ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE ENGINEERING AND TECHNOLOGY

EVOLUTIONARY DESIGN ASSISTANTS FOR ARCHITECTURE

Ph.D. THESIS

N. Onur SÖNMEZ

Architecture Department

Architectural Design Graduate Programme

1 MAY 2015

ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE ENGINEERING AND TECHNOLOGY

EVOLUTIONARY DESIGN ASSISTANTS FOR ARCHITECTURE

Ph.D. THESIS

N. Onur SÖNMEZ
(502042006)

Architecture Department

Architectural Design Graduate Programme

Thesis Advisor: Prof. Dr. Arzu ERDEM
Co-advisor: Prof. Dr. Sevil SARIYILDIZ

1 MAY 2015

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

MİMARLIK İÇİN EVRİMSEL TASARIM ASİSTANLARI

DOKTORA TEZİ

N. Onur SÖNMEZ
(502042006)

Mimarlık Anabilim Dalı

Mimari Tasarım Yüksek Lisans ve Doktora Programı

Tez Danışmanı: Prof. Dr. Arzu ERDEM
Eş danışman: Prof. Dr. Sevil SARIYILDIZ

1 MAYIS 2015

N. Onur SÖNMEZ, a **Ph.D.** student of **ITU Graduate School of Science Engineering and Technology,** student ID **502042006,** and **TU Delft Graduate School for Architecture and the Built Environment**, on a joint supervision of the PhD program, successfully defended the **thesis** entitled "**EVOLUTIONARY DESIGN ASSISTANTS FOR ARCHITECTURE**", which he prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor :	**Prof. Dr. Arzu ERDEM** İstanbul Technical University	.....
Co-advisor :	**Prof.Dr.Ir. Sevil SARIYILDIZ** Delft University of Technology	.....
Jury Members :	**Dr. Ir. Rudi STOUFFS** Delft University of Technology	.....
	Prof. Dr. Belkıs ULUOĞLU İstanbul Technical University	.....
	Prof. Dr. Ir. Imre HORVATH Delft University of Technology	.....
	Prof. Dr. Ir. Bauke de VRIES Eindhoven University of Technology	.....
	Prof. Dr. Jose P. DUARTE University of Lisbon	.....

Date of Submission : 20 February 2015
Date of Defense : 1 May 2015

FOREWORD

I wish to express my sincere appreciations and thanks to my thesis supervisors; Arzu ERDEM, who has constantly motivated and guided me in my research efforts and supported me on all kinds of problems encountered within a PhD process, with a tireless creativity and problem-solving energy; and Sevil SARIYILDIZ, who trusted me for the joint supervision program and undertook the supervision of my thesis, with her considerate support and guidance through the hardships of the process.
I should also express my deep gratitude and appreciation to Rudi STOUFFS, not only for his patient reviews and invaluable advice that enabled me to move through the versions of the thesis draft, but in the first place, for his generous help that enabled my visit to TU Delft as a researcher, which has dramatically changed the direction of the study.
I would also like to thank Belkıs ULUOĞLU and Birgül ÇOLAKOĞLU for their kind attendance and advices at my review committee meetings; Gülen ÇAĞDAŞ and Hakan TONG for providing me with the inspiration and enthusiasm through their courses, which led me to prepare the first draft of a project that would eventually evolve into a PhD thesis; and Bige TUNÇER for academic support and advice while I stayed in Delft.
I should thank several colleagues for their important help: Ioannis CHATZIKONSTANTINOU, Ali PAŞAOĞLU, Figen IŞIKER, and Asım DİVLELİ for their help during the preparation and tutoring of the two thesis workshops and Sema ALAÇAM for her gracious help in the preparation of a review committee meeting. Amongst my colleagues, I should especially thank Ahu SÖKMENOĞLU for many things at once: for providing crucial references and contacts, for her help in the tutoring of the İzmir workshop, but most of all, for being a companion all through this, at times, overwhelming process.
This research was made possible by the financial support of TUBITAK (The Scientific and Technical Research Council of Turkey) and Istanbul Technical University Scientific Research and Development Support Program. Yet, I also feel obliged to thank a series of heros of the open source movement, who present their support for free; in particular, the communities that created and maintained the Python language and useful packages Scipy, Numpy, Matplotlib, Cairo, Shapely, and Polygon; open source applications, Inkscape, Blender, and Gimp; and the whole community of people who share their knowledge and resources through the internet just for the joy of sharing.
I owe many thanks to my academic and non-academic friends, who enabled me to retain my mental stability at least long enough to finish this thesis. Among these friends, special thanks have to be dedicated to Onur KESKİN, who listened to my troubles sometimes for hours, and without a single complaint.
For sure, the real actualizers of this thesis have been my parents, Emine and Kadircan SÖNMEZ, who apparently desired the title of PhD next to my name quite a

bit more than myself. I should also not forget the support and care of my sister Fatma and brother in law Şener YÜZSEVEN, my nephew Kaya, who expanded my mental horizons with his dinasour researches, and my niece Nil.

May 2010 N. Onur SÖNMEZ

TABLE OF CONTENTS

Page

FOREWORD ..... **vii**
TABLE OF CONTENTS ..... **ix**
ABBREVIATIONS ..... **xiii**
LIST OF TABLES ..... **xv**
LIST OF FIGURES ..... **xvii**
SUMMARY ..... **xxiii**
ÖZET ..... **xxv**
1. INTRODUCTION **1**
1.1 Problem and Motivation 1
1.1.1 The draft making assistant 1
1.1.2 Artificial / Autonomous / Automated Design 3
1.1.3 A clash between two paradigms 8
1.2 Aims and Research Questions 11
1.3 Research Method 14
1.4 Overview of the Thesis 18
2. DESIGN AND COMPUTATION: A RESEARCH PROGRAM FOR ARTIFICIAL / AUTONOMOUS / AUTOMATED DESIGN **23**
2.1 Design as Ability, Activity, and Profession 26
2.1.1 Design research 28
2.1.2 Design as purposeful transformation and planning: the problem-solving paradigm 33
2.1.3 Design is solution focused: co-evolution and bridging 40
2.1.4 Design as conversation with a situation: reflection-in-action 43
2.1.5 Design as inquiry into a unique problem situation 46
2.1.6 Phases of design and decomposition of design processes 47
2.1.7 Models for the design realm 49
2.2 Illustration by story-telling: the library problem 53
2.2.1 The problem: design problems tend to be vague 54
2.2.2 Moving on: framing, proposing, negotiating 55
2.2.3 Different viewpoints 57
2.2.4 Design actions change the situation 58
2.2.5 Precedents and the co-evolution of problems and solutions 58
2.2.6 Dynamic utilization of prefabricated strategies 59
2.2.7 Contextuality 60
2.2.8 Constraints, objectives, and criteria 61
2.2.9 Dynamic structuring strategies 62
2.2.10 A multitude of possibly conflicting objectives 63
2.3 Computational Design 65
2.3.1 The extended mind and the human-tool complex 73
2.3.2 Computational Design as a research field 76

2.3.3 Artificial intelligence in design: information-processing, problem-solving, search, and formal approaches 77
2.4 Challenges of Design vs. Human Strategies 86
2.4.1 Design by search and on-line exploration 87
2.4.2 Complexity of design situations 93
2.4.3 Ubiquity of interpretative skills 96
2.4.4 Problem of (re)framing and an expertise built over everyday intelligence 97
2.4.5 Exploratory journey with the help of mental maps 97
2.4.6 Design knowledge and rigor: Design Methods Movement 99
2.4.7 Dynamic focus and continuous (re)framing 101
2.4.8 Integral approaches 104
2.4.9 Dynamic decomposition and integration 105
2.4.10 Different kinds of performances 107
2.5 The Research Program of Artificial / Autonomous / Automated Design 113
2.6 design_proxy: An Integrated Approach for Draft Making Design Assistants 125
2.7 Conclusion 129
3. EVOLUTIONARY COMPUTATION FOR DESIGN 133
3.1 What is Evolutionary Computation? 133
3.2 Brief History of Evolutionary Computation 136
3.3 Evolutionary Computation in Art, Design, and Architecture 138
3.4 Basic Evolutionary Algorithm Issues 141
3.5 Evolutionary Computation for Design: Requirements, Advantages, Drawbacks 144
3.6 Complex and Hierarchical Evolutionary Algorithms 151
3.7 Interleaved EA: Multi-Objective Evolutionary Computation for Design 158
3.8 Conclusion 164
4. APPLICATIONS OF THE DESIGN PROXY APPROACH AND THE INTERLEAVED EA 167
4.1 design_proxy for Graphics 167
4.1.1 Task definition 167
4.1.2 A review of related studies 169
4.1.3 Visual interface 171
4.1.4 Representation, initiation, evaluation, operators, and the evolutionary process 172
4.1.5 Application cases for d_p.graphics and the naive Interleaved EA 188
4.1.6 Single-objective tests and adaptivity 189
4.1.7 Multi-objective tests for abundant and regular cases 194
4.1.8 Naive Interleaved EA vs. Pareto-based version 198
4.1.9 Conclusions and an evolutionary collaboration model 203
4.2 Situating Evolutionary Computation within Architectural Design: Architectural Stem Cells Framework and design_proxy.layout 206
4.2.1 Problem Structuring and Architectural Stem Cells Framework 208
4.2.2 Layout problem and a review of related studies 221
4.2.3 design_proxy.layout 227
4.2.4 Task definition, representation, initiation, selection, variation, and the evolutionary process 234
4.2.5 Evaluation 244
4.2.6 Test series and verification 251
4.2.7 One and two objective test series 253

4.2.8 Multi-objective test series 268
4.2.9 Validation of the design_proxy.layout 278
4.2.10 Evolutionary Collectivity İzmir workshop 280
4.2.11 Evolutionary Collectivity Mardin workshop 299
4.2.12 Overall evaluation of the design_proxy.layout 309
4.3 Conclusion 316
5. CONCLUSIONS AND FUTURE RECOMMENDATIONS 321
5.1 design_proxy Approach 323
5.2 Interleaved Evolutionary Algorithm 323
5.3 design_proxy.layout 326
5.4 Architectural Stem Cells Framework 329
5.5 Theoretical Outputs and Additional Remarks 331
REFERENCES 337
APPENDICES 351
CURRICULUM VITAE 359

ABBREVIATIONS

2D : Two-dimensional
3D : Three-dimensional
AD, A^3D : Automated / Autonomous / Artificial Design
ADI : Artificial Design Intelligence
AI : Artificial Intelligence
AMG : Architecture Machine Group
ASC : Architectural Stem Cell
BIM : Building Information Modeling
BREEAM : Building Research Establishment Environmental Assessment Method
CAD : Computer Aided Design
CAAD : Computer Aided Architectural Design
CBIR : Content-Based Image Retrieval
CBSR : Content-Based Shape Retrieval
d_p : design_proxy
d_p.graphics : design_proxy.graphics
d_p.layout : design_proxy.layout
DNA : Deoxyribonucleic Acid
DU : Design Unit
EA : Evolutionary Algorithm
EC : Evolutionary Computation
ES : Evolution Strategies
EP : Evolutionary programming
FBS : Function Behavior Structure
GA : Genetic Algorithm
GP : Genetic Programming
HVAC : Heating, Ventilation, and Air Conditioning
IEA, Interleaved EA: Interleaved Evolutionary Algorithm
LEED : Leadership in Energy and Environmental Design
PLD : Plan layout design
SEED : Software Environment to Support Early Phases of Building Design

LIST OF TABLES

Page

Table 2.1 : Ontological categories in situated FBS ontology..... 91
Table 2.2 : Mapping of fundamental steps and activities in situated FBS ontology. 91
Table 4.1 : Evolutionary Collectivity workshop survey. 313

LIST OF FIGURES

Page

Figure 1.1 : Three main methodological threads of the study. 15
Figure 1.2 : Basic research instruments of the study. 18
Figure 2.1 : Problem-solving task used in the experimentation by Alexiou, Zamenopoulos and Johnson (2009). 37
Figure 2.2 : Design task used in the experimentation by Alexiou, Zamenopoulos, and Johnson (2009). 38
Figure 2.3 : The co-evolution model of Maher, Poon, and Boulanger (1996)..... 43
Figure 2.4 : Analysis – synthesis – evaluation model of Lawson (2005, p. 49). 48
Figure 2.5 : The model of design problems via constraints (Lawson, 2005, p. 106). 50
Figure 2.6 : Model of design activities (Lawson and Dorst, 2009, p. 51). 52
Figure 2.7 : Levels of design activities (Lawson and Dorst, 2009, p. 61). 53
Figure 2.8 : Basic search concepts. 80
Figure 2.9 : General scheme of generate and test algorithms. 80
Figure 2.10 : Local search by 'greedy hill-climbing'..... 82
Figure 2.11 : Evolutionary parallel search. 82
Figure 2.12 : Offline algorithm. 83
Figure 2.13 : Online algorithm. 83
Figure 2.14 : External, interpreted, and expected function-behavior-structure representation of design process (Gero and Kannengeisser, 2007). 89
Figure 2.15 : Basic constituents of a design situation. 94
Figure 2.16 : A summary of the basic problems of design situations (left side) and the strategies and skills of human designers (right side). 112
Figure 2.17 : Basic constituents of the design_proxy approach. 126
Figure 3.1 : Satellite dish holder boom; left, initial regular design; right, geometry of the final optimized design (Keane and Brown, 1996). 138
Figure 3.2 : An exploratory task structure for evolutionary design. 150
Figure 3.3 : Problem structuring (decomposition and integration) through productions..... 152
Figure 3.4 : Adaptive evolution. 153
Figure 3.5 : Multi-level evolutionary search as a learning system. 156
Figure 3.6 : Schematic illustration of an Interleaved EA process..... 160
Figure 3.7 : Basic structure of the Interleaved EA. 160
Figure 3.8 : Potential parallelization of design_proxy-based assistants. 164
Figure 4.1 : Graphic arrangement task. 167
Figure 4.2 : Genotype for d_p.graphics. 172
Figure 4.3 : Initiation (Desktop 1 scenario). 173
Figure 4.4 : Mutation operators..... 176
Figure 4.5 : Penalty calculations for Out + Overlap objective. 177
Figure 4.6 : Cell, row, and column preparation for Cell and Sequence objectives. 177
Figure 4.7 : Color fitness calculations. 178
Figure 4.8 : Basic parameters of the d_p.graphics application. 180

Figure 4.9 : Evolutionary process for the self-adaptive Interleaved EA for the d_p.graphics application. 183
Figure 4.10 : Adaptive determination of the current objective. 185
Figure 4.11 : Modifications in case of stagnation (for d_p.graphics implementation). 186
Figure 4.12 : Single objective test series through the desktop wallpaper scenario; 10 runs of 200 generations for each objective. Non-adaptive and process-adaptive versions are given, both use the best-so-far initial parameters. 191
Figure 4.13 : Comparison of non-adaptive and self-adaptive versions with best parameter settings. Single-objective runs for Cell objective. Each graph is the average of 10 runs. 193
Figure 4.14 : Comparison of non-adaptive and self-adaptive versions with poor parameter settings. Single-objective runs for Cell objective. Each graph is the average of 10 runs. 194
Figure 4.15 : Inputs and results of the first Coffee-cup scenario, regular and abundant (upper right) cases. 195
Figure 4.16 : Examples of multi-objective runs: Coffee-cup 2 scenario and its abundant version (process-adaptive). 196
Figure 4.17 : Results of the Desktop 1 scenario, non-abundant versions (process-adaptive). 197
Figure 4.18 : Results of the Desktop 2 scenario, abundant and non-abundant versions together (process-adaptive). 197
Figure 4.19 : Results of the T-shirt 1 scenario, abundant and non-abundant versions together (process-adaptive). 197
Figure 4.20 : Results of the T-shirt 2 scenario, abundant and non-abundant versions together (process-adaptive). 198
Figure 4.21 : Comparison chart for average performances of different version combinations. 199
Figure 4.22 : Example processes and results for the Pareto-based version (left) and self-adaptive naive Interleaved EA (rightmost column). 201
Figure 4.23 : The effect of the initiation maps. 203
Figure 4.24 : Collaboration model for Interleaved EA. 205
Figure 4.25 : Levels for a hierarchical decomposition of architecture. 208
Figure 4.26 : A layer-based task decomposition approach. 209
Figure 4.27 : (a) A traditional decomposition of a mobile robot control system into functional modules, (b) A decomposition of a mobile robot control system based on task achieving behaviors (Brooks, 1999). 210
Figure 4.28 : A layer based task decomposition scheme for conceptual architectural design. 214
Figure 4.29 : An abstract 3D formation as an architectural stem cell. 217
Figure 4.30 : Three example ASCs, manually developed within the task-based decomposition framework. 218
Figure 4.31 : Example target, ground floor of Santa Monica Library by Moore Ruble Yudell Architects. Vector painted on Inkscape according to given layers and swatches (seen on the right). 232
Figure 4.32 : Example target simplified (only DUs), ground floor of Santa Monica Library. 233
Figure 4.33 : Library functions. In each column, painting layers (left side) is mapped to a simplified library program (right side). 234

Figure 4.34 : Basics of problem representation. 234
Figure 4.35 : Layout generation for multiple floors with arbitrary outlines. 235
Figure 4.36 : Candidate preparation and initiation. Left: painted candidate, middle: initiated candidate, right: best result of evolution. 239
Figure 4.37 : Mutation operators. 241
Figure 4.38 : Interleaved EA process for d_p.layout. 242
Figure 4.39 : Modification of the objective priority list for d_p.layout. 244
Figure 4.40 : Calculation of the Out + Overlap fitness. 245
Figure 4.41 : Calculation of the Proximity fitness. 246
Figure 4.42 : Calculation of the Trivial Hole fitness. 247
Figure 4.43 : Calculation of the Sequence fitness. 248
Figure 4.44 : Calculation of the Neighbor fitness. 249
Figure 4.45 : Calculation of the Neighbor Cell fitness. 250
Figure 4.46 : Calculation of the Manual Cell fitness. 251
Figure 4.47 : Trivial Hole objective, for 50 DUs. 255
Figure 4.48 : Trivial Hole objective, for 150 DUs. 256
Figure 4.49 : Trivial Hole objective, for 250 DUs. 257
Figure 4.50 : Target and candidate (tight fit) for Sequence objective. 258
Figure 4.51 : Control evolution with only Trivial Hole objective (tight fit scenario). 258
Figure 4.52 : Sequence objective verification – Sequence + Trivial Hole, tight fit candidate, best results of 0° and 0° + 45° versions – Sequence grid width: 2 meters. 259
Figure 4.53 : Sequence + Trivial Hole, example fitness graphs. 260
Figure 4.54 : Sequence objective verification – Sequence and Out + Overlap, loose fit candidate, best results of 0° and 0° + 45° versions – Sequence grid width: 2 m. 260
Figure 4.55 : Sequence and Out + Overlap, example fitness graphs. 261
Figure 4.56 : Target, candidate, and initiation for Sequence objective. 261
Figure 4.57 : Random control runs with only Trivial Hole objective. 261
Figure 4.58 : Sequence objective verification – Sequence and Out + Overlap – Sequence grid width: 2 m. 262
Figure 4.59 : Sequence and Out + Overlap, example fitness graphs – sequences [3, 4, 5], rotations [0°, 45°]. 263
Figure 4.60 : Neighbor and Adjacency objectives; target, candidate, and initiation. 263
Figure 4.61 : Neighbor and Adjacency objectives, example best results – DU offset: 1.2 m. 264
Figure 4.62 : Neighbor + Trivial Hole, example fitness graphs. 265
Figure 4.63 : Adjacency + Trivial Hole, example fitness graphs. 265
Figure 4.64 : Neighbor Cell and Adjacency Cell objectives; target, candidate, and initiation. 266
Figure 4.65 : Neighbor Cell and Adjacency Cell objectives, example best results – Query cell width: 8 m. 266
Figure 4.66 : Neighbor Cell + Trivial Hole, example fitness graphs. 267
Figure 4.67 : Adjacency Cell + Trivial Hole, example fitness graphs. 267
Figure 4.68 : Manual Cell objective; target, candidate, and initiation. 267
Figure 4.69 : Manual Cell objective, example best results – Grid width: 4m, query cell width: 8m. 268
Figure 4.70 : Manual Cell + Trivial Hole, example fitness graphs. 268

Figure 4.71 : Average mean fitness graphs for the multi-objective case – 7 scenarios, 100 runs for each – 6 objectives, 50 DUs, population: 50, mutation: 5, crossover: 0. 270
Figure 4.72 : Prepared outlines and fixed DUs of a five-floor example run. 273
Figure 4.73 : Fitness graphs for the ground floor of the example run. 274
Figure 4.74 : Resulting best floor layouts for the example run (max. gen.: 10000, pop.: 150, cross.: 0, mut.: 15, Manual Cell target (("vc", "t", "t", "t", "wc", "wc")), Neighbor Target = Neighbor Cell target: Santa Monica, Sequence target: Free University Berlin, using max. %92 of the available plan areas, DU offset: 1.2m, Sequence grid width: 1.5m, M. Cell grid width: 8m, query cell widths: 16m). 276
Figure 4.75 : Workshop process for Evolutionary Collectivity İzmir workshop. 281
Figure 4.76 : Images from the Evolutionary Collectivity İzmir workshop, from brainstorming sessions to final presentations. 282
Figure 4.77 : Simplified color-coding legend. 286
Figure 4.78 : Müge and Ceren – left: initial sketches, middle: first model, right: sketch submitted for d_p.layout. 287
Figure 4.79 : Müge and Ceren, results of the first trial (seven floors, max. gen.: 7000, pop.: 150, cross.: 0, mut.: 15, Manual Cell target (("vc", "t", "t", "t", "wc", "wc")), Neighbor Target = Neighbor Cell target: Santa Monica, Sequence target: Free University Berlin, using max. %92 of the available plan areas, DU offset: 1.2m, Sequence grid width: 1.5m, M. Cell grid width: 8m, query cell widths: 16m, DU area limit: 45m^2, DU width limit: 8m, formal refinement on with mut.:75, diminish: 25) 288
Figure 4.80 : Müge and Ceren, solid-void exercise over d_p.layout drafts. 289
Figure 4.81 : Müge and Ceren, sketches submitted for second d_p.layout iteration. 290
Figure 4.82 : Müge and Ceren, results of the second iteration (six floors, max. gen.: 8000, DU area limit: 30m^2, DU width limit: 4.5m). 291
Figure 4.83 : Müge and Ceren, final proposal. 292
Figure 4.84 : İdil Kantarcı's first d_p.layout trial, as prepared in Inkscape by the tutors. 293
Figure 4.85 : İdil's first d_p.layout results (DU area limit: 48m^2, DU width limit: 8m) 294
Figure 4.86 : Left: İdil's solid-void exercise for two floors, right: preparation for the second trial. 295
Figure 4.87 : İdil's second trial. Top: initial submission printed after being prepared and parsed. Bottom: outputs of the trial in their original layouts (DU area limit: 50m^2, DU width limit: 8m). 296
Figure 4.88 : İdil's sketches from the third day. 297
Figure 4.89 : İdil Kantarcı's final product (top-left: ground floor, top-right: first floor over the ground floor). 298
Figure 4.90 : Photographs from the Evolutionary Collectivity Mardin workshop. 300
Figure 4.91 : Workshop process for the Evolutionary Collectivity Mardin workshop. 301
Figure 4.92 : Büşra Tekelioğlu and Merve Çap, top-left: initial massing, right: ground floor of d_p.layout output (DU area limit: 130m^2, DU width limit: 10m), bottom-left: solid-void models. 302

Figure 4.93 : Second day sketches of Büşra and Merve, from d_p.layout drafts, toward final models. 303
Figure 4.94 : Büşra and Merve, final model (bottom-left: ground floor, bottom-right: first floor over ground floor). 304
Figure 4.95 : Merve, Bahri, and Melike, conceptual sections. 305
Figure 4.96 : Merve, Bahri, and Melike, d_p.layout preparation and results (DU area limit: 100m^2, DU width limit: 8m). 305
Figure 4.97 : Merve, Bahri, and Melike, solid – void studies and the first envelope proposal. 306
Figure 4.98 : Merve, Bahri, and Melike, sketches toward the final model. 306
Figure 4.99 : Merve, Bahri, and Melike, final proposal. 308
Figure 4.100 : Survey results (mean) for Izmir and Mardin workshops. 314
Figure 4.101 : design_proxy approach and its application for design_proxy.layout. 317
Figure 4.102 : Basic constitution of the ASC Framework. 318
Figure A.1 : Library of "Universitad Catolia del Norte". 352
Figure A.2 : Library for the Faculty of Philology, Free University of Berlin. 353
Figure A.3 : Halmstad Library. 354
Figure A.4 : Santa Monica Library. 355
Figure A.5 : Troyes Mediatheque. 356
Figure A.6 : TU Delft Library. 357

EVOLUTIONARY DESIGN ASSISTANTS FOR ARCHITECTURE

SUMMARY

In its parallel pursuit of an increased competitivity for design offices and more pleasurable and easier workflows for designers, artificial design intelligence is a technical, intellectual, and political challenge. While human-machine cooperation has become commonplace through Computer Aided Design (CAD) tools, a more improved collaboration and better support appear possible only through an endeavor into a kind of artificial design intelligence, which is more sensitive to the human perception of affairs.

Considered as part of the broader Computational Design studies, the research program of this quest can be called Artificial / Autonomous / Automated Design (AD). The current available level of Artificial Intelligence (AI) for design is limited and a viable aim for current AD would be to develop design assistants that are capable of producing drafts for various design tasks. Thus, the overall aim of this thesis is the development of approaches, techniques, and tools towards artificial design assistants that offer a capability for generating drafts for sub-tasks within design processes. The main technology explored for this aim is Evolutionary Computation (EC), and the target design domain is architecture. The two connected research questions of the study concern, first, the investigation of the ways to develop an architectural design assistant, and secondly, the utilization of EC for the development of such assistants.

While developing approaches, techniques, and computational tools for such an assistant, the study also carries out a broad theoretical investigation into the main problems, challenges, and requirements towards such assistants on a rather overall level. Therefore, the research is shaped as a parallel investigation of three main threads interwoven along several levels, moving from a more general level to specific applications. The three research threads comprise, first, theoretical discussions and speculations with regard to both existing literature and the proposals and applications of the thesis; secondly, proposals for descriptive and prescriptive models, mappings, summary illustrations, task structures, decomposition schemes, and integratory frameworks; and finally, experimental applications of these proposals. This tripartite progression allows an evaluation of each proposal both conceptually and practically; thereby, enabling a progressive improvement of the understanding regarding the research question, while producing concrete outputs on the way. Besides theoretical and interpretative examinations, the thesis investigates its subject through a set of practical and speculative proposals, which function as both research instruments and the outputs of the study.

The first main output of the study is the "design_proxy" approach (d_p), which is an integrated approach for draft making design assistants. It is an outcome of both theoretical examinations and experimental applications, and proposes an integration of, (1) flexible and relaxed task definitions and representations (instead of strict formalisms), (2) intuitive interfaces that make use of usual design media, (3) evaluation of solution proposals through their similarity to given examples, and (4) a

dynamic evolutionary approach for solution generation. The design_proxy approach may be useful for AD researchers that aim at developing practical design assistants, as has been examined and demonstrated with the two applications, i.e., design_proxy.graphics and design_proxy.layout.

The second main output, the "Interleaved Evolutionary Algorithm" (IEA, or Interleaved EA) is a novel evolutionary algorithm proposed and used as the underlying generative mechanism of design_proxy-based design assistants. The Interleaved EA is a dynamic, adaptive, and multi-objective EA, in which one of the objectives leads the evolution until its fitness progression stagnates; in the sense that the settings and fitness values of this objective is used for most evolutionary decisions. In this way, the Interleaved EA enables the use of different settings and operators for each of the objectives within an overall task, which would be the same for all objectives in a regular multi-objective EA. This property gives the algorithm a modular structure, which offers an improvable method for the utilization of domain-specific knowledge for each sub-task, i.e., objective. The Interleaved EA can be used by Evolutionary Computation (EC) researchers and by practitioners who employ EC for their tasks.

As a third main output, the "Architectural Stem Cells Framework" is a conceptual framework for architectural design assistants. It proposes a dynamic and multi-layered method for combining a set of design assistants for larger tasks in architectural design. The first component of the framework is a layer-based, parallel task decomposition approach, which aims at obtaining a dynamic parallelization of sub-tasks within a more complicated problem. The second component of the framework is a conception for the development mechanisms for building drafts, i.e., Architectural Stem Cells (ASC). An ASC can be conceived as a semantically marked geometric structure, which contains the information that specifies the possibilities and constraints for how an abstract building may develop from an undetailed stage to a fully developed building draft. ASCs are required for re-integrating the separated task layers of an architectural problem through solution-based development. The ASC Framework brings together many of the ideas of this thesis for a practical research agenda and it is presented to the AD researchers in architecture.

Finally, the "design_proxy.layout" (d_p.layout) is an architectural layout design assistant based on the design_proxy approach and the IEA. The system uses a relaxed problem definition (producing draft layouts) and a flexible layout representation that permits the overlapping of design units and boundaries. User interaction with the system is carried out through intuitive 2D graphics and the functional evaluations are performed by measuring the similarity of a proposal to existing layouts. Functioning in an integrated manner, these properties make the system a practicable and enjoying design assistant, which was demonstrated through two workshop cases. The d_p.layout is a versatile and robust layout design assistant that can be used by architects in their design processes.

MİMARLIK İÇİN EVRİMSEL TASARIM ASİSTANLARI

ÖZET

Tasarım ofisleri için daha yüksek rekabet kapasitesi ve tasarımcılar için ise daha kolay ve keyifli iş akışlarının geliştirilmesi yolunda ikili bir araştırma gündemini takip ederken, Yapay Tasarım Zekâsı araştırmacıları hem teknik, hem entelektüel, hem de politik meydan okumalar karşısında kalırlar. Her ne kadar insan - makine işbirliği Bilgisayar Destekli Tasarım (CAD) araç ve süreçleri üzerinden hatırı sayılır bir yaygınlık kazanmışsa da, mevcut olanın daha ötesinde bir işbirliği ve daha iyi tasarım desteği ancak insanların dünya kavrayışlarına karşı daha hassas ve kavrayışlı olabilen bir tür Yapay Zekânın geliştirilmeye başlanmasıyla mümkün olabilecektir. Hesaplamalı Tasarım (Computational Design) araştırmalarının bir parçası olarak düşünülebilecek bu araştırma alanı Yapay / Otonom / Otomatik Tasarım olarak adlandırılabilir. Tasarım için kullanılabilen Yapay Zekâ düzeyi hâlihazırda sınırlı olduğu için, güncel Yapay Tasarım çalışmaları için makul bir hedef, çeşitli tasarım görevleri için taslak tasarımlar üretmek olabilir. Dolayısıyla, bu tez çalışmasının genel ölçekteki hedefi, tasarım süreçlerindeki alt görevler için taslak tasarımlar üretebilecek yapay tasarım asistanlarına yönelik yaklaşımlar, yöntemler ve araçlar geliştirmek olmuştur.

Bu amaçla keşfedilmeye çalışılan teknoloji Evrimsel Hesaplamalar, hedef alınan tasarım disiplini ise mimarlık olarak belirlenmiştir. Çalışmanın birbiriyle bağlantılı iki araştırma sorusunun ilki, bir yapay mimari tasarım asistanının nasıl geliştirileceğinin soruşturulmasını, ikincisi ise bu tür asistanların üretiminde Evrimsel Hesaplamaların nasıl kullanılabileceğinin araştırılmasını konu almıştır. Çalışma, böylesi bir asistan için yaklaşım, yöntem ve hesaplamalı araçlar geliştirirken, aynı zamanda geniş bir kuramsal inceleme gerçekleştirmeyi de hedeflemiştir. Bu kuramsal inceleme, konuyla ilgili temel problemler, güçlükler ve gerekliliklerin daha genel bir düzeyde anlaşılmasını sağlamaya yöneliktir.

Araştırma üç paralel araştırma hattının iç içe geçtiği çeşitli düzeylere yayılmıştır. Bu üç araştırma hattının birincisi, hem mevcut kuramsal yazına dönük hem de çalışmanın öneri ve uygulamalarına dönük kuramsal inceleme, tartışma ve kurgusal öneriler üzerinden takip edilmiştir. İkinci araştırma hattı hem betimleyici, hem yöntem öneren modeller, haritalamalar, özet görselleştirmeler, görev yapıları, görev ayrıştırma şemaları ve bütünleştirici çerçeveler üretmeye yönelmiştir. Üçüncü araştırma hattı ise tüm bu önerilerin deneysel uygulamalarını kapsar. Genel bir kuramsal düzeyden tekil uygulamalar düzeyine kadar yayılan bu üç hat araştırma süreci boyunca eşzamanlı olarak, bir arada işlenmiş ve geliştirilmiştir. Bu üç yönlü ilerleyiş her önerinin hem kavramsal açıdan hem de pratikte değerlendirilmesini, dolayısıyla araştırma sorularına yönelik kavrayışın aşamalı olarak gelişmesini sağlamış, aynı zamanda da süreç boyunca somut çıktılar ortaya koymuştur. Başka deyişle, kuramsal ve yorumsal incelemelerin yanısıra, çalışma, ele aldığı konuyu bir seri, uygulamaya dönük veya kurgusal öneri yoluyla da araştırmıştır. Bu öneriler tez çalışmasının hem araştırma araçları, hem de çıktılarıdır.

Çalışmanın ilk ana çıktısı "tasarım_vekili" (design_proxy, d_p) adı verilen ve tasarım taslakları üreten asistanlar geliştirmeye yönelen bütüncül bir yaklaşımdır. Yaklaşım hem kuramsal incelemelerin hem de deneysel uygulamaların bir sonucu olarak ortaya çıkmıştır. tasarım_vekili yaklaşımı, (1) esnek ve gevşetilmiş görev tarif ve temsillerinin (katı formelleştirmeler yerine), (2) alışılageldik tasarım araç ve ortamlarının kullanıldığı sezgisel ve doğrudan arayüzlerin, (3) çözüm önerilerinin verili örneklere benzerlikleri üzerinden otomatik değerlendirilmesinin ve (4) ürün geliştirmek için dinamik bir evrimsel yaklaşımın bütünleşik olarak kullanılmasını önerir. Yaklaşımın işlerliği tezde yer alan iki uygulama üzerinden araştırılmış ve gösterilmiştir. tasarım_vekili yaklaşımı Yapay Tasarım araştırmacılarının yorum ve kullanımlarına sunulmuştur.

Tezin ikinci ana çıktısı, "İçiçe Evrimsel Algoritma" (Interleaved Evolutionary Algorithm, Interleaved EA, IEA) adı verilen yeni bir evrimsel algoritma türüdür. Bu algoritma tasarım_vekili yaklaşımına dayanan tasarım asistanlarının üretken mekanizması olarak kullanılmak üzere geliştirilmiştir. İçiçe Evrimsel Algoritmanın işleyişinde sürecin alt hedeflerinden biri zindelik gelişmesi duraklayana kadar tüm süreci yönlendirir. Burada yönlendirmekten kastedilen, evrimsel kararlarda genel olarak sürece ait değil de lider hedefe has zindelik değerlendirmelerinin ve varyasyon ayar ve operatörlerinin tüm süreci yönlendirmekte kullanılabilmesidir. Lider hedef kurgusuna sahip olmayan evrimsel algoritmalarda ise sürece ait ayarlar ve değerlendirmeler ortak olarak kullanılmaktadır. Bu haliyle IEA genel bir süreçteki her bir farklı alt hedef için ayrı ayarlar ve operatörler kullanılmasını mümkün kılmaktadır ve dinamik ve uyarlanabilir bir çoklu-hedefli evrimsel algoritma olarak işlev görürken aynı zamanda tekil hedefli evrim süreçlerine ait bazı yöntemlerin kullanılabilmesini de mümkün kılmaktadır. Bu özellikleri IEA'nın modüler bir yapıya sahip olmasını sağlar ve uygulama alanına has bilginin her bir alt görev (hedef) temelinde sürece katılmasını sağlama potansiyelini açığa çıkarır. IEA Evrimsel Hesaplamalar araştırmacıları ve Evrimsel Hesaplamalardan yararlanan uygulamacılar tarafından kullanılabilir.

Tezin üçüncü ana çıktısı, "Mimari Kök Hücreler Çerçevesi" (Architectural Stem Cells Framework), mimari tasarım asistanlarına yönelik kavramsal bir çerçevedir. Bu çerçeve bir seri tasarım asistanının daha üst bir mimari tasarım görevi için bir arada işleyebilmesi yolunda çoklu-katmanlı ve dinamik bir yöntem önerir. Çerçevenin ilk bileşeni, katman-tabanlı bir paralel görev-ayrıştırma yaklaşımıdır –ki daha karmaşık bir problemin içindeki alt-görevlerin dinamik bir paralelizasyonunu hedefler. Çerçevenin ikinci bileşeni bina taslakları geliştirmeye yönelik mekanizmalara dair bir kavramlaştırmadır. Bu kavramlaştırmaya "Mimari Kök Hücreler" adı verilmiştir. Bir Mimari Kök Hücre semantik olarak işaretlenmiş bir geometrik yapı olarak anlaşılabilir. Bu yapı soyut bir binanın, kısmen tanımlanmış bir problem bağlamında, detaysız bir durumdan tümüyle gelişmiş bir bina taslağına doğru nasıl gelişebileceğine yönelik olasılık ve sınırlamaların bilgisini barındırır. Mimari Kök Hücre kurgusu çözüm-tabanlı bir üretim sürecinde birbirinden ayrıştırılacak olan alt görev katmanlarının nasıl yeniden bütünleştirileceğine yönelik bir yanıt olarak gerekli görülmüştür. Dolayısıyla, çerçevenin ilk bileşeni dinamik görev-ayrıştırma, ikinci bileşeni ise esnek bir yeniden-bütünleştirme yapısını konu edinmiş, çerçeve bu iki bileşenin nasıl bir arada işleyeceğine yönelik kurgusal bir öneri sunmayı hedeflemiştir. Bu haliyle, Mimari Kök Hücreler Çerçevesi tezin ortaya attığı pek çok fikir ve öneriyi pratiğe yönelik geniş bir araştırma gündemi içinde bir araya getirmeyi hedeflemiştir ve mimarlık alanındaki Yapay Tasarım araştırmacılarına sunulmuştur.

Tezin sonuncu çıktısı, "tasarım_vekili.plan_şeması" (design_proxy.layout, d_p.layout) adı verilen mimari plan şeması tasarım asistanıdır. Bu asistan, tasarım_vekili yaklaşımına dayanmaktadır ve önerilerini geliştirmek için İçiçe Evrimsel Algoritma'yı kullanır. tasarım_vekili yaklaşımının bir sonucu ve uygulaması olarak, sistem, gevşetilmiş bir problem tarifi (taslak plan şemaları üretmek) ve plan birimlerinin üst üste çakışmasına ve plan sınırlarından taşmasına izin veren esnek bir plan şeması temsili kullanmaktadır. Kullanıcı etkileşimi alışılageldik grafik tasarım araçları ve çıktıları üzerinden sağlanmakta, otomatik işlevsel değerlendirmelerde ise mevcut planlar ile öneri planlar arasındaki karşılaştırmalardan elde edilen zindelik değerleri kullanılmaktadır. Belirli bir alt-hedef açısından hedef alınan bir binaya daha çok benzeyen önerinin zindeliği daha yüksektir. Bu temel özellikler bütünleşik bir yolda işleyerek sistemi pratikte kullanılabilir ve keyifli bir tasarım asistanı haline getirmiştir. Bu iddialar tezde iki öğrenci atölyesi üzerinden desteklenmektedir. tasarım_vekili.plan_şeması asistanı çok yönlü, süreçteki hata ve eksiklikleri tolere edebilen son derece esnek bir plan taslağı asistanıdır ve mimarlar ve mimarlık öğrencileri tarafından tasarım süreçlerinin özellikle erken aşamalarında kullanılabilir.

1. INTRODUCTION

1.1 Problem and Motivation

1.1.1 The draft making assistant

In one of the scenes of the documentary, "Sketches of Frank Gehry" (Pollack, 2006), the well-known senior architect, Gehry, gives instructions to "his architect", to cut cardboard pieces and stick these over each other to obtain a building form. How gratuitous these sketch models feel when they are being built. Yet, after each move, the genius gains an opportunity to re-evaluate the situation, and orders another move. Consider also snapshots from architectural offices that we got used to after 2000s, a group of architects discussing in front of a table filled with massing alternatives cut from polystyrene foam. What happens in both cases is the integrated usage of brainstorming and sensitive, multifaceted evaluation.

It is true that the design moves that generate or develop alternatives or proposals sometimes appear gratuitous, and there might really be a degree of randomness in these moves. However, randomness is only one side of the process. For first, none of the moves is completely random. Rather, these moves involve a degree of randomness limited by a complex interaction of weakly guiding rationale, which are not always possible to explicate. Secondly, after a move is carried out, it becomes possible to evaluate the result with regard to an arbitrary number of aspects; apparently, sometimes just with respect to the whims of a senior designer, which nevertheless hide a well-internalized manner of carrying out tasks.

A combination of the above-mentioned cases reveals how design progresses through proposals; through the bringing forward of proposals with regard to a set of rationale, to be evaluated after being observed within a multi-layered and changing context. Design involves preparing drafts to be evaluated, revised, and refined. This is what was described by Schön (1983) as the 'see-move-see' cycles. This mechanism is an essential part of design, yet it is not always explicitly discerned.

Although brainstorming (freely generating a set of options to be critically examined later) is not sufficient on its own to complete a design task, it has been the most widespread and long lasting remnant of the Design Methods Movement, i.e., the first generation of design research. As a design method, brainstorming can be interpreted as corresponding to the 'move' step of the fundamental 'see-move-see' mechanism of design. It allows scarcely elaborated ideas that are not well thought out on many aspects to come to surface, so that it becomes possible to see a range of available options. Only after this partially aleatory move, that is, only after the positive act of draft making, it becomes possible to move forward through the evaluation of the new situation. This is also the essence of sketching, which can be seen as a complex interweaving and integration of brainstorming and evaluation. In general, this is the explanation of solution-based thinking, which is the basic design strategy for the structuring of a design situation, through which both the problem and the solution start to emerge.

At this point, it becomes intriguing to ask, whether an artificial design agent could assume the role of the architect of Frank Gehry, or if it could ease the burden of the ascetic novices of a design office. Consider a sketch for an artificial design agent: This agent should have a capacity to function within an architectural design office, amongst a series of human designers and other office workers. Just like the human architects, this agent has to deal with design tasks through its set of styles, or manners of carrying out its tasks. What kind of artificial intelligence (AI) approaches and systems would this agent use? What would be its level of success in dealing with its problems? What will be the problems that this agent would be dealing with? What kind of information and knowledge would it embody or solicit, and how? What would we gain from such agents in design? Which tasks in design could such an agent assume? In Gehry's case, elaborate draft formations for integrated cladding and massing. In the latter case, alternative massing studies with regard to weakly considered functional and contextual issues.

Such draft developing functionality, which apparently does not require a human level intelligence, may be a starting point for the development of artificial design intelligence. Could intelligent artificial agents help us, while we are searching, retrieving, classifying, and studying our precedents; while we are developing an architectural proposal; while we are sketching or drawing? Could it function as a

tool, as an assistant, as an apprentice, or as a partner? Could it cause us to be more productive, more creative, more visionary, cleverer, or happier designers?

1.1.2 Artificial / Autonomous / Automated Design

In this thesis, the research program that aims at developing methods for intelligent tools and assistants will be called Artificial / Autonomous / Automated Design (AD, or perhaps A3D), and will be considered as an offshoot of AI studies in general. AI is the traditional umbrella term attributed to the fields that seek to develop methods, tools, and agents that would undertake tasks that are acknowledged as requiring a form of intelligence. The product of such research has also been called AI.

The definition of intelligence unavoidably varies through time, worldviews, tasks, and contexts. Thus instead of trying to attain a simplistic or overly generalized consensus on this definition, in the context of this study, intelligence will be understood in a rather practical sense and in its essential plurality. In the case of AD, intelligence is understood with regard to design tasks, hence in the form of artificial design intelligence.

Before moving further, we should clarify what we mean by AD in comparison with the phrases, Computer Aided Design (CAD) and Computational Design. Traditionally, the term CAD has been used (1) to denote a group of computer tools for design related tasks, (2) new design workflows that have started to appear with the advent of these tools, (3) a research field that develops these tools, and (4) a line of business that produces and markets such tools. These four references should be separated. In this thesis, the first sense of CAD will be termed as CAD tools or CAD systems. The term CAD will be used to refer to the second sense, i.e., any design process that benefits computer aids. The third sense of CAD will be understood as a part of Computational Design, and will be referred to as Computational Design. Computational Design, before denoting a type of design practice or process, is a field of research, which searches for better and more advanced CAD, where CAD refers to all design processes that utilize computer tools. While the frontier of the field is constantly being shifted, after being absorbed within daily practice, basic usage of a computational technique or tool is no more seen as part of research (unless it is a learning system). It is the development of computational tools and methods, rather than designing with these tools, which distinguishes the field of Computational

Design. For instance, developing a computational method for design may be considered within Computational Design. If this method was developed and used as part of a design process, i.e., within a research-by-design process, the process is both CAD and Computational Design. If the developed method becomes a tool or a technique that is regularly being used for design, it becomes a CAD tool and the design efforts that utilize this tool can simply be called CAD; however, mere use of this method is no longer Computational Design.

AD should be considered as a subfield of Computational Design. However, it is difficult to delineate the borders of AD within Computational Design studies. It appears that there is rather a continuity between techniques that are considered more or less intelligent. The classification axis adopted in this study attempts at determining the level of the operational intelligence of a system with regard to its manner of operation, i.e., according to whether it operates, (1) as a tool, which must be instructed line by line, (2) as an assistant, which can take instructions and carry out several steps, or (3) as intelligent design agents, which should be completely autonomous (Kalay, 2004, p. 419).

As distinct from some areas of Computational Design, AD does not target one-off methodical innovations, but only reusable strategies, techniques, methods, and tools. This means that reusable techniques may be added to the CAD tools repository. Thus, the short-term practical aim of AD is a more advanced CAD, while the theoretical and intellectual aim will remain a better understanding of design intelligence, its challenges, methods, and capabilities. This will also be a path to a higher degree of automation. Intelligent artificial agents will aim at automating a larger chunk of human territory; however, it might also be the case that the artificial and human collaborators together enlarge the design field, without spilling over into each other's domains.

As a sub-area of CAD, Computer Aided Architectural Design (CAAD) has a broad range of tools in its service, which has enhanced architectural design and allowed formal and technical advancements in built artifacts. However, the development of partially or fully autonomous intelligent design agents has remained a challenge for creative tasks within architectural design. Studies on intelligent technologies for architecture (i.e., AD) have commenced at around the same period with the first instances of architectural visualization tools, which are usually thought as rather less

intelligent. Both research paths gained considerable attention from researchers, yet architectural drawing and modeling tools have long become commonplace, while AD staggered. So, it should be asked, why has it not been possible to develop intelligent artificial agents for architectural design, after around fifty years of studies? Not only a comprehensive practical system is lacking, autonomous artificial agents do not exist for even apparently simple tasks.

Yet, we could also ask, what is a simple task in architecture? Deciding on where to locate a mass within a site? Generating a series of massing studies for a building? Arranging a series of functional zones with respect to contextual, psychological, functional, and stylistic aspects? Deciding on the color of a wall? Choosing materials for finishing a floor? Negotiating with a series of stakeholders? Discussing with a team of designers?

It is clear that these are not simple tasks. They all require an understanding of a series of complicated issues, such as the complex superpositions and interactions of a series of building systems; cultural, political, and formal aspects of an urban context; the history and progression of cultural codes and formal styles; the multifarious requirements within a problem; perceptual and ergonomic aspects of spaces; and the dynamic structuring of the extremely complicated process of design itself. None of these issues has a predefined and fixed textbook specification. On the contrary, these are contextual, open-ended, and dynamic in character; even worse, they are dependent on each other. Development of intelligent design agents is difficult, because design is complicated. Even defining a design task is a problematic issue, because of the non-linear and open-ended character of the co-evolution of the problems and the solutions.

This has apparently led most investors and developers to focus on less intelligent CAD tools and mostly leave artificial design intelligence issues to a group of enthusiastic researchers. Does this mean that artificial design intelligence would not help architects in carrying out their tasks? Not at all, as can be seen through the highly specific techniques of AI that have already been embedded within CAD tools. 3D modeling and rendering tools extend the computational reach of their users by helping them in complex calculations that they would or could not attempt to carry out, if these had not been automated. The extension made possible by these tools is not limited to representation. In digital modeling and surveying Building Information

Modeling (BIM) technologies are rapidly developing. These systems automate the propagation of design moves towards semantically labeled building elements, while at the same time enabling automated surveying of quantities. Likewise, simulation tools for air conditioning, daylighting, acoustics, and emergency evacuation achieve an extension of computational capabilities, both in terms of speed and scope. With its analysis tools that come closer to the region of artificial design intelligence, Space Syntax research claims to have a degree of success in automated spatial analysis. Furthermore, it can be claimed that a certain threshold has been reached in several types of CAD tools. Representational tools, drawing, modeling, and graphic design applications are now reliable and productive.

Because intelligent support for creative aspects of design has been relatively neglected, a potential expansion area for CAD systems appear as incorporating more developed artificial design intelligence for better and more productive human-tool interactions, not only for detailing and optimization phases, but for creative and conceptual tasks as well; as these are the distinguishing aspects of architectural design in comparison with engineering design fields.

We function together with computer systems today. We live as human-tool assemblages. It can be expected that the development of interfaces for the heterogeneous relationships between experts and non-experts, between humans and tools, and between different non-human systems will continue to be a challenge that is appropriate for AD research. Just as CAD tools increased the productivity, sensitivity, and capabilities of the design practices, more automation could support this improvement in the same direction. However, the simple assumption "the more intelligent and capable the machine counterpart the better the assemblage" has to be revised to take into account the collective functioning of the human and tool counterparts: "the more collaborative the machine counterpart, the better the assemblage". Better collaboration, better support appears possible only through an endeavor into an intelligence, which is sensitive to the human perception of affairs.

In brief, artificial design intelligence is at first a technical challenge, which has a potential for useful results. At the same time, it is an intellectual challenge, a frontier, shifting the boundary of the well-understood productive processes. Finally, it is political, in being related to the humans' unending struggle against the toil of compulsory work. From a rather professional viewpoint, intelligent design tools

could mean affordable, competitive, innovative, better-integrated, higher quality architectural services. Assuming roles such as representative media, external memory, production tools, and collaboration platforms, CAD tools have already started to change the profession in this direction. However, with the acceleration of the construction business and because of the ambitious demands raised by these developments, a large percentage of architects is today filling the ranks of a white-collar precariat with never-ending work hours; instead of moving towards the aspired direction for becoming well-paid and well-respected experts who reach out for the greater good of the public.

With the advent of the creative economy debates, better workspaces and working conditions became more significant aspects in the corporate world. Such values have started to be perceived as more than the empty desires or fantasies of marginals or daydreamers. They became legitimate demands in even business life, as epitomized in the Google Company's headquarter buildings. Better workspaces and working conditions are both a task and a requirement for design, as design is a business, too. Not only inhabited spaces but also organizations and workflows have effects on these issues.

At the same time, innovation and creativity came to be treated as the most important competitive assets. These demand time, effort, intelligence, and longer work-hours, although at the same time they have a potential to make work more exciting and satisfying. Nevertheless, 'better process' could indicate not only creative, fast, and adaptive, but also equitable, lateral, pleasurable, and participatory design processes that could satisfy all stakeholders including the designers themselves. It should be remembered that even without a promise of increases in productivity and competitivity, these values are worthy of respect and following on their own.

Although it might not be possible to predict the exact outcome of such an endeavor, the author of this thesis nevertheless has a propensity towards the emancipatory potential of automation in all fields, which takes the form of AD research in architecture. The main orientation of this thesis has been situated over an ideological stance, which involves an equitable profession in the form of horizontal collaborations, weaker hierarchies, decreased exploitation, increased knowledge sharing, effective participation, transparency, and wider accessibility; in short, a more horizontal and rewarding production regime. It is not claimed that these would

be obtained solely through the results of this thesis. Obviously, it is not possible to discuss these issues through the meager proposals of this thesis. Nevertheless, it is still intriguing to ask, if AD could provide more pleasurable and easier workflows, shorter work hours, and more satisfying processes to the ever-exploited young architects, while simultaneously increasing the competitivity of the design offices.

1.1.3 A clash between two paradigms

While an increased intelligence and better collaboration appear as the basic requirements of artificial design intelligence, an engineering-oriented paradigm has been dominating the studies in related areas. This paradigm has resulted in an oversimplified comprehension of design situations within the frameworks of engineering disciplines. Research is mostly oriented towards the more engineering compliant aspects and tasks in design, leaving creative aspects aside. Another tendency has been to redefine design tasks in the form of mathematical or engineering problems, i.e., within the problem-solving paradigm, which omits the real complications of design situations. This has made progress difficult in AD, and the resulting failure helped endure the mystical aura of design, at least within the designer community.

The engineering-oriented approach, which is part of a broader paradigm of technical rationality, prioritizes formal and methodological rigor and practical applicability to the detriment of design, in the sense that it makes researchers half blind to the core characteristics of the fields like architectural design. Even worse, the very formation of the academic fields of architecture forces researchers to stand on either side of a deep chasm, where two incompatible viewpoints create a parallax effect. On one side, there is design research and theory, which, following the reflection-in-action conception of Donald Schön, yields only overall interpretative knowledge, hard to defend in terms of the criteria of the technically oriented perspective, or to utilize within practical schemes. Although there are intermediary research paths such as protocol studies, design theory approaches more towards the humanities and the social sciences. On the other side, there is the dominance of an engineering and mathematics oriented technical rationality, which pursues applied research processes, targeting definite and defendable products. In reality, most of these products are not as useful as it is often claimed. The products are brought forward, formalized,

defended, verified, validated, etc.; yet, most of these systems are never accepted into practice. Such research tends to take care of methodological requirements more than the requirements of the practice, which is nevertheless understandable. Because, in reality, these studies become useful only by being added to the lineages of larger research programs; through the methods and ideas that they bring forward, develop, improve, or hint at; or through interpretations that they make possible, which is indeed parallel with what happens in the more ambiguous, less product-oriented research approaches.

As a result of the oversimplified comprehension of design, the engineering view insists on apprehending design situations in terms of engineering problems, to the extent that their most important characteristics are 'pruned'. The result is a proliferation of research projects that tend to endure the domination of a reductive view towards design, devoid of sensitive understanding. Eventually, the above-mentioned chasm results in the designers' tendency to oscillate between the two sides, according to whether they are the critics of a research project or the developers. When designers assume the role of the project developers or researchers, they are forced to present products that have to be defended in terms of the methodologies of applied fields, dominated by an engineering paradigm. On the other hand, when they are the critics, designers pretty much understand that the reductive path, which usually demands the critic to be highly forgiving to the shortcomings of a project, is not that innocent. Simplification of real-world problems and tasks is often a practical necessity in research projects; however, each act of simplification runs the risk of becoming a reduction of the real complexity, of the very aspects to be confronted.

AD demands an attitude, which adopts both sides of the chasm: an awareness of social science approaches, a propensity towards humanities, a hacker sensibility within practice (because the field has always remained in an age of the beginnings), but also the methodological rigor of applied sciences; a rigor that is to be assumed for practical aims, not as a worldview. Rigor or formalization cannot be the initial goal of AD, but may be put to use where appropriate. Formal definitions and mathematical rigor are the basis of, for example, the underlying mathematical operations of architectural drawing and modeling applications. They are indeed the basis of all computer technology. However, such formalizations are not sufficient as

the primary providers of the principles of artificial design agents. Such principles may only be obtained through design research.

Design typically involves complicated, multifaceted, and ill-defined situations, which are still challenges for AI studies. Design intelligence is not inferior to a mechanical, calculative intelligence; on the contrary, it requires a more complicated kind of intelligence. One aspect of such intelligence is everyday understanding, while another is the expert knowledge within specific fields. Everyday intelligence still defies AI techniques, and mundane interpretation tasks for humans often correspond to the so-called AI-hard, or AI-complete problems. Given the necessity to judge functional, economic, cultural, and aesthetic aspects at any time throughout design processes, dealing with most design procedures require an understanding of the physical world, daily life, and culture, in addition to expert skills. Therefore, it will be claimed that, design is AI-complete, in other words, design problems are amongst the hardest AI problems[1]. For this reason, developments in AD would help advance AI in general. It is not surprising that there is a disbelief regarding design automation and a secondary aim of this study is to provide a detailed assessment of the complexity of design situations in order to investigate why design is AI-complete and on which paths we could tackle this challenge.

Advancements in AI usually proceed in parallel with the advancements in cognitive sciences; new AI approaches give way to new cognitive models, hence to a better understanding of human thinking. Design is a special mental activity, through which some of the most complicated real world problems are being tackled. AD may contribute to a better understanding of the design activity; thus, may help cognitive studies in general.

Design is highly context-dependent. This is why design methods or tools cannot be defined as context-free formal schemes or models. A design problem has to be re-interpreted within each particular situation, throughout a parallel evolution of problems and solutions, where an additional non-linear dimension arises because of the simultaneous transformation of the contextual aspects. Therefore, the difficult tasks in the quest for AD are, first, attaining the interpretative intelligence, capable of

[1] Hanna (2012) referred to this situation as "the hard problem of design".

appreciating daily situations, and secondly, gaining the capability for the dynamic structuring of a design process in line with the changes in the interpretation of a design situation. The lack of such mechanisms is the basic reason for the failure of the first generation AI approaches (sometimes called the GOFAI, i.e., the good old-fashioned AI, in a rather pejorative manner) in AD. Nevertheless, AI continues to advance with new methods, while a probabilistic, statistical, and example-based intelligence, which has to function in a situated, real-time, and on-line manner, is gaining prominence. The challenge continues and the current question is, what other technologies are available for AD? This is the current frontier for AD research.

1.2 Aims and Research Questions

Despite growing success of practical AI applications, a complete automation for architectural design tasks is not a practical target for today. While current techniques are not sufficient for completely autonomous artificial design agents, it is an underlying contention of this thesis that artificial design intelligence may still be pursued on other levels of intelligence, i.e., on the level of design assistants. A twofold progression is required for this aim. The first path orientates us towards readily useful design tools or assistants, and it can be termed as "weak-AD". The second path has a long-term perspective towards a complete automation of design tasks, and can be called "strong-AD". In this study, we will move over the first (practical) path, with a background awareness of the second as the ultimate target of AD. This means, bringing together a practical research with a broader theoretical perspective.

Given the state of the art in AI studies, for practical AD approaches, collaboration between human agents and artificial systems is inevitable. CAD takes place in a design environment where the humans and CAD tools operate together. The question is devising approaches that would enable a gradual increase in the capabilities of the artificial assistants. At one point, this question should involve artificial systems that are able to learn the knowledge and skills that expert human designers possess. This can only be achieved if appropriate collaboration strategies are developed. Therefore, the current task of AD may be described as developing artificial design assistants that are not only useful but also intuitive and pleasant to use. A potential function of such assistants could be the production of draft designs for various sub-tasks within design

processes. These drafts could in turn be used by human designers as a means to contemplate on alternatives, or as intermediary points to be revised and improved.

With the stating of this aim, another question arises: what are the potential AI technologies for such assistants? While the criticisms against earlier AI approaches were becoming commonplace, "embodied cognition" and "situated AI" approaches came to target these deficiencies by internalizing most of these criticisms. At the same time, a group of computation techniques called "soft computing" had been on the rise. Neural Networks, Ant Colony Optimization, Simulated Annealing, and Evolutionary Computation (EC)—all of which take their lessons from natural or material processes—are within this group. Amongst these techniques, as a local stochastic search method, EC has almost become the paradigm for problem-solving (see Michalewicz and Fogel, 2004).

In design fields, EC has been utilized for a variety of aims, most importantly within performance-oriented approaches, where the evolutionary component is mainly used as an optimization tool. However, it is not compulsory to understand EC as an optimization tool. EC has found various applications within different fields and conceptual frameworks. For example, within the field of artificial life, evolution is understood as a complex adaptation mechanism. Within Genetic Programming (GP), evolutionary mechanisms are used for evolving—or learning—computer programs or machine-learning models. There have been numerous applications of evolutionary techniques for arts and music for automated generation of images and musical pieces. Beside the evolutionary mechanisms used for these applications, a shared objective has been to automate a task, which had previously been carried out predominantly by humans. From another viewpoint, these studies explore potential roles for EC in the development of artificial agents, which can take over a portion of the workload of the humans, or at least help them carry out their tasks in an easier and more effective manner.

The basic mechanism of an evolutionary algorithm (EA) cycles through proposing, evaluating, and modifying a set of alternatives, which is reminiscent of the see-move-see mechanism of design, although on a much simpler level. Nonetheless, operating such a mechanism on a more elaborate level would require a more improved and multifaceted artificial design intelligence, which is not available. The operating level of the EC mechanism appears compatible with the basic available

level of AI for design, which is limited to a set of task representation, generation, and evaluation methods. Therefore EC appears as a viable candidate as the underlying generative technology of a draft making assistant.

This study, thus, limits its task by setting out EC as the hypothetical basis for the development of design assistants. The basic task that arises from this statement concerns locating EC within design fields and identifying its potentials as well as limitations. How could EC be exploited for the generation of artificial design assistants? What are its current limitations and how could these limitations be alleviated with other approaches and techniques? Utilization of machine-learning approaches would be another important aspect of the above-mentioned draft making assistant. However, the breadth of the task compelled us to limit the practical study to the basic assistant that could 'just work' within a regular design environment. The potential of using EC for learning systems will be examined as part of the study; however, other machine-learning approaches will not be examined and they will only be indicated as potential development paths.

On an overall level, the thesis involves design in general, while on a particular level, the thesis is mainly concerned with architectural design. Therefore, the task will be further limited by gradually focusing on architectural design.

In brief, the aim of this thesis is the development of approaches and techniques towards artificial design assistants that offer a capability for developing drafts for sub-tasks within design processes. This aim requires a broad investigation into the main problems, challenges, and requirements towards such assistants, through both theoretical investigations and practical applications. The technology to be explored for this aim will be Evolutionary Computation (EC), and target design domain will be architecture.

In conclusion, there are two connected research questions:

1. How can we develop an architectural design assistant for draft-making purposes?

2. Can Evolutionary Computation (EC) be utilized for the development of draft making assistants for architectural design?

The integrated research question becomes:

How can we develop an evolutionary architectural design assistant for draft-making purposes?

1.3 Research Method

It is obvious that human designers manage to handle complicated design problems. However, despite widespread consensus on several important issues, there is no textbook on how to carry out design. While it is assumed that an amount of knowledge is being held tacitly within the minds of the human designers and implicitly within design products, explicitly stated procedural knowledge is rare. In other words, design theory does not have a firm prescriptive dimension. This is because, design concerns complicated problems and strategies, which are usually peculiar to specific situations. Design actions and interpretations are highly context-dependent and individual designers tend to develop their own ways of carrying out design. These make design a vast, under-structured field. As a result, an inquiry towards an aspect of this field requires an exploratory approach, which aims at structuring its subject while inquiring into it. Even in its limited state, the subject of this thesis concerns a broad area, which involves design practice and theory, Computational Design, and EC studies. The question is how to investigate this broad subject, and the answer is to bring together a series of different methodological approaches as interrelated threads of exploration.

This research is shaped as a parallel investigation of three main threads that are interwoven through the progression of the study (Figure 1.1). These threads involve, first, theoretical examinations, discussions, and speculations with regard to both existing literature and the proposals and applications of the thesis; secondly, proposals for descriptive and prescriptive models, mappings, summary illustrations, task structures, decomposition schemes, and integratory frameworks; and finally, experimental applications of these proposals.

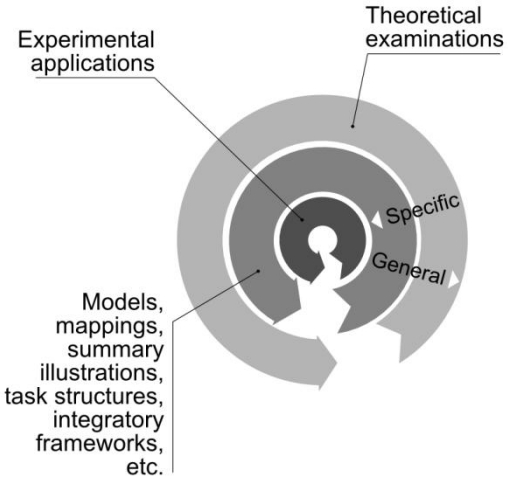


Figure 1.1 : Three main methodological threads of the study.

Such tripartite progression allows an evaluation of each proposal both conceptually and practically in terms of its weaknesses and limitations on the one hand, and strengths and comprehensiveness on the other; thereby enabling a progressive improvement of the understanding regarding the research question, while producing concrete outputs on the way, from the most general theoretical level to the most particular application level.

The multitude of discussions, models, illustrations, frameworks, and applications can be understood as probes into design situations. Here, the phrase 'design situation' refers to the task environment of a design agent. These probes can also be understood as viewpoints that are directed at different scales, aspects, and fields within a broad area. Each probe may involve a different method as required by its goals. Such an exploration will inevitably span a multitude of levels of generality. We can support this methodology through a conceptualization of perspectivism, according to which no single viewpoint would yield a sufficient understanding towards such a complicated subject. Each viewpoint potentially generates a different kind of narrative, as a function of its specific targets, scale of generality, limits, details, manner of description, etc.; in short, as a consequence of where, why, and how it looks. These viewpoints may be situated within distinct levels of generality, but they may also be complementary or alternative positions on the same level. All these perspectives may partially overlap or contradict with each other; nevertheless, in most of the cases, several of them are required for a comprehensive understanding.

Besides theoretical and interpretative examination, the thesis investigates its subject through a set of practical and speculative proposals (Figure 1.2), which function as both research instruments and the outcomes of the study:

- The "**design_proxy**" approach (d_p):

  An integrated approach for draft making design assistants. It is an outcome of both theoretical examinations and experimental applications and proposes an integration of, (1) flexible and relaxed task definitions and representations (instead of strict formalisms), (2) intuitive interfaces that make use of usual design media, (3) evaluation of solution proposals through their similarity to given examples, and (4) a dynamic evolutionary approach for solution generation. The design_proxy approach may be useful for AD researchers that aim at developing practical design assistants.

- The "**Interleaved Evolutionary Algorithm**" (IEA, or Interleaved EA):

  A novel evolutionary algorithm, proposed and used as the underlying generative mechanism of the design_proxy-based design assistants. The Interleaved EA is a dynamic, adaptive, and multi-objective EA in which one of the objectives leads the evolution until its fitness progression stagnates. It leads the evolution in the sense that the settings and fitness values of this objective is used for most evolutionary decisions. In this way, the Interleaved EA enables the use of different settings and operators for each of the objectives within an overall task, which would be the same for all objectives in a regular multi-objective EA. This property gives the algorithm a modular structure, which offers an improvable method for the utilization of domain-specific knowledge for each sub-task, i.e., objective. The Interleaved EA can be used by Evolutionary Computation (EC) researchers and by practitioners who employ EC for their tasks.

- The "**Architectural Stem Cells**" (ASC) Framework:

  A conceptual framework for artificial architectural design assistants. It proposes a dynamic and multi-layered method for combining a set of design assistants for larger tasks in architectural design. The first component of the framework is a layer-based, parallel task decomposition approach, which aims at obtaining a dynamic parallelization of sub-tasks within a more

complicated problem. The second component of the framework is a conception for the development mechanisms for building drafts, i.e., Architectural Stem Cells (ASC). An ASC can be conceived as a semantically marked geometric structure, which contains the information that specifies the possibilities and constraints for how an abstract building may develop from an undetailed stage to a fully developed building draft. ASCs are required for re-integrating the separated task layers of an architectural problem through solution-based development. The ASC Framework brings together many of the ideas of this thesis for a practical research agenda and it is presented to the AD researchers in architecture.

- "**design_proxy.graphics**" (d_p.graphics) application:

  A graphic design assistant for a toy task, mainly used for testing and demonstrating the use of design_proxy approach together with the Interleaved EA.

- "**design_proxy.layout**" (d_p.layout) application:

  An architectural layout design assistant based on the design_proxy approach and the Interleaved EA. The system uses a relaxed problem definition (producing draft layouts) and a flexible layout representation that permits the overlapping of design units and boundaries. User interaction with the system is carried out through intuitive 2D graphics and the functional evaluations are performed by measuring the similarity of a proposal to existing layouts. Functioning in an integrated manner, these properties make the system a practicable and enjoying design assistant, which was demonstrated through two workshop cases. The d_p.layout is a versatile and robust layout design assistant that can be used by architects in their design processes.

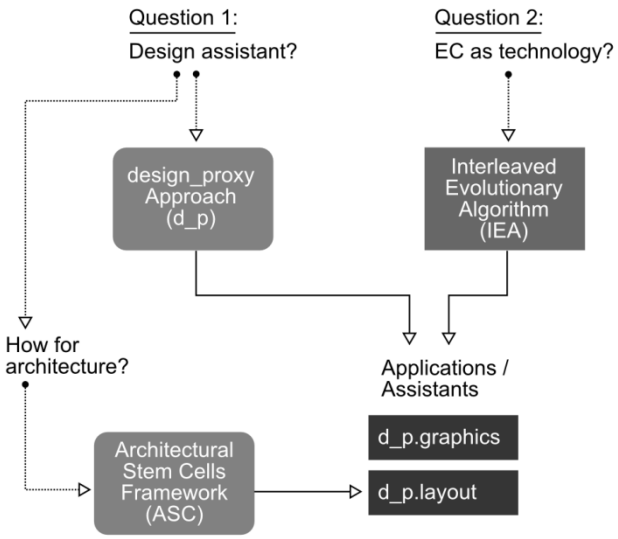


Figure 1.2 : Basic research instruments of the study.

1.4 Overview of the Thesis

The thesis starts on an overall, theoretical level, which involves critical theoretical investigations into design theory and Computational Design for comprehensively examining why previous approaches have had problems in responding to the quest for artificial design intelligence. The practical goal is determining a viable research program for Artificial / Autonomous / Automated Design (AD). Thus, Chapter 2 is dedicated to a detailed theoretical outline, concerning the relationships of design processes and Computational Design, which at the same time reveals, supports, and discusses the aims and directions adopted in the thesis.

The first part of the chapter attempts at presenting a mature conception of design processes by examining what design is and how it is carried out by humans; through descriptions of the complex, vague, undefined, contextual, conflict-laden, and negotiatory character of design situations and corresponding dynamic, non-linear, solution-based, and co-evolutionary process structuring efforts of human designers. Basic outputs of this examination are a mapping of the basic constituents of design situations and a classification of the multifarious evaluations required within these.

The second main subject of Chapter 2 involves Computational Design, Computer-Aided Design (CAD), and Artificial Intelligence (AI) in design. Through a critical examination of the history of Computational Design studies, potential AI strategies, current limitations, and required intelligence types and capabilities for artificial

design agents are identified. Another basic output of this investigation is a mapping of the difficulties encountered by the AI studies in design as linked to the corresponding strategies and techniques employed by human designers. The expectation is to develop a better understanding on potential research paths for AD.

By way of these investigations and discussions, a broad research program is determined for prospective studies, over which Evolutionary Computation (EC) could be grafted. This research program is based on a practical conceptual basis for human-machine relationships (i.e., weak-AD), yet its ultimate aim will remain comprehensive (i.e., strong) AD. As an outcome of this chapter, the design_proxy (d_p) approach will be presented as an integrated strategy towards draft making assistants. The main tenets of this approach acquire their rationale from the broad research program of AD. The specific proposals of this thesis and their practical results will be situated over this research program and following the design_proxy approach, which is envisaged to bring together a heterogeneous set of techniques and approaches for a series of different design tasks. For each new application, the aspects of the specific application are to be gathered under a specific title.

The third chapter of the thesis concerns Evolutionary Computation (EC) and its usage in design in general (that is, not in a specific field). The overall research task for this level concerns the examination of EC and its usage in design, by focusing on its limitations, potentials, and requirements through theoretical investigations. After an overall introduction to the basics of EC, a brief history of EC and its usage examples will be introduced, and the basic issues, requirements, advantages, and drawbacks of EC will be detailed. There is a set of more definite research questions on this level, which are to be examined through different methods. The first of these questions concerns the determination of how EC would be located within design processes and in what kind of collaboration (agency) scheme. Thus, the first practical output of the chapter is a task structure for EC use in design tasks. The second question inquires for the specificities of EAs that are required for complicated design problems. Therefore, in combination with the above-mentioned task structure, several schemes for complex and hierarchical EAs will be brought forward.

The last section of Chapter 3 introduces the "Interleaved EA", which is a dynamic, complex, adaptive, and multi-objective evolutionary algorithm (EA) devised specifically for design. The discussions on EC usage for design and the presentation

of the Interleaved EA constitute the first part of an answer to the research question regarding the utilization of EC for AD. As for the second part of the answer, i.e., for a concrete utilization of this complex EA instance within specific design problems, the two applications of the design_proxy approach will be presented in detail in Chapter 4.

First main section of Chapter 4 is involved with the illustration of the application and testing of the design_proxy approach and the Interleaved EA in a graphic design context. The approaches, methods, preconditions, and underlying evolutionary mechanisms will be demonstrated alongside the verificatory tests for some functionalities of the system. The d_p.graphics application has been used for developing, studying, and evaluating the Interleaved EA over a series of axes such as versatility through adaptivity, evaluation through similarity, an easy to adapt graphical interface, and a collaboratory information and knowledge gathering framework. These techniques constitute the basis for the second application of the thesis (i.e., the design_proxy.layout).

The second main section of Chapter 4 concerns the development of an EC based assistant for the creative tasks of architectural design (i.e., the design_proxy.layout). There are several preliminary steps before arriving at a specific application for an architectural task. The initial overall question is how creative, conceptual architectural design may be structured to constitute a basis for an artificial design assistant. The proposals for structuring architectural design situations through a layer based decomposition scheme and the Architectural Stem Cells (ASC) Framework are presented and discussed as part of the initiatory steps. Because design tasks are dynamically intermingled in real design processes, the structuring attempted with these proposals will not be based on a temporal progression or phases, but rather on a dynamic parallelization of task layers. These proposals for structuring the vaguely delimited field of architectural design appear as a specific research framework for computational architectural design, which can set forth new paths and potentials for future studies. As such, they enable this study to situate the architectural layout task within a broader agenda, and to locate the d_p.layout system within the general context of architectural design.

With a background aim towards strong-AD, the research program assumed by this thesis can only be possible with the admission of the developed artificial systems into

real world design processes. Such a system should find a way to be either useful, or pleasurable, or both. Therefore, the last research task for the thesis is to investigate the capability of d_p.layout to be adopted within design situations. This would also function as an experiment for the viability of the above-mentioned research program. A secondary target is the investigation of the collaborative potential of d_p.layout. Thus, the sub-sections from 4.2.9 to 4.2.12 are dedicated to the illustration and discussion of the usage of the system during two student workshops.

There are complex relationships between the different levels and outputs of this thesis, due to a simultaneous development process. Most of the approaches developed for the thesis appear open for further development. Therefore, following an examination of the basic outputs of the thesis, the concluding chapter will also include a discussion of the outcome of each research thread, and additionally, starting with the potential additions and revisions on the proposed systems, will try to set forward a series of future research paths.

2. DESIGN AND COMPUTATION: A RESEARCH PROGRAM FOR ARTIFICIAL / AUTONOMOUS / AUTOMATED DESIGN

The overall aim of this chapter is to draw a detailed theoretical outline for the aims and directions adopted in this thesis. The investigation will start with a broad examination of design. This examination will help us to develop a mature conception of design processes, so that we can better comprehend potential research paths over which EC could be grafted. The second concern of this chapter will be computer-aided design (CAD), Computational Design, and artificial intelligence (AI) in design. The main aim in studying these issues is to base the practical proposals of the thesis over a critical examination of previous Computational Design studies. It should be noted that, the theoretical examination will not end with this chapter. Rather, it will start in this chapter and will continue to develop through considerations on proposals and applications in the following chapters.

The first part of this chapter focuses on the description of design, fields and activities considered under the term, and several descriptive models on separate aspects of this activity. Additionally, several key terms that concern this thesis will be introduced. It should be stressed that this will not be an attempt on an all-encompassing discussion of existing views. Rather, the aim is to present a specific understanding towards design, as a ground for the analysis of design activity that will be advanced throughout the thesis. It is not a goal of this thesis to develop a new definition for design. Besides its redundancy, this would require a completely different manner of research, well out of the bounds of this study. Therefore, instead of bringing out new research, the following effort of understanding will be supported by a series of well-established texts within design research.

As such, with an aim to develop a mental framework, a basis, a ground for the proposals of the thesis, the general concept of design will be explored on three levels. The examination will start on a most general level with the broad question "What is design?" Moving one level down, a more detailed description for design will be given through the plurality of activities of different design fields. Design fields

exhibit important differentiations in their practices and approaches. Thus, a definition that would encompass most of these fields should be kept on an appropriate abstraction level. At this point, an important distinction emerges between design as a common ability and design as an expertise, which enables us to set forth basic issues regarding design knowledge. On the third level, through the progression of a discussion regarding a series of views on design (i.e., design as problem-solving, design as a reflective practice, and design as inquiry into a situation) several important terms such as problems, moves, and rationale will be introduced. On the last level, three models will be cited from prominent design theorists Bryan Lawson and Kees Dorst, whose views are amongst the most important influences on the viewpoint adopted within this study. These models can be considered sound and well established and are able to present some crucial aspects of design clearly and succinctly. While the number of cited models could have been increased, these three models are envisioned as the smallest package of complementary models that would together delineate a unified image for design. An additional criterion for the selection was the models' relevance for the specific exploratory path of this study, in the sense that they will be further referred to for the proposals and products of this thesis.

Until this point, the investigation exhibits the character of a collection of existing concepts, ideas, and models, which are more or less abstract. However, because practical design knowledge is mostly tacit and cannot be expressed through generalized, context-free propositions, the design situation would best be described through design cases, i.e. within context. Thus, the abstract descriptions of the preceding sections will be further detailed and illustrated with a design story depicting an imaginary architectural design process, with an aim to visualize these conceptions within a common frame. The story is imaginary, yet it will be based on the conceptions extracted from existing design research. While the story unfolds, several issues regarding the complexity of architectural problems will be illustrated, e.g., inherently vague, undefined, contextual, conflict-laden, negotiatory character of design situations and corresponding dynamic, non-linear, solution-based, co-evolutionary process structuring efforts of human designers.

After this introduction into design situations, the examination will proceed towards Computational Design, firstly, with an aim to develop a practical conceptual basis for human-machine relationships within production processes, and secondly, to set forth

a broad and plausible research program for prospective studies. The text will go on to define Computational Design primarily as a research field, and will try to anticipate a plausible path of progression while determining basic assumptions and characteristics of such a path. Starting with the meaning of computation, terms such as Computational Design, CAD, and AI will be discussed with respect to a human-tool collectivity encountered throughout the history, by way of an extension of human mind and capabilities into the environment. Toward a critical examination of the AI studies in design, basic constituents of the information-processing paradigm will be introduced with its corollaries, problem-solving, search, and formal approaches. This introduction will also establish a series of connections important for the understanding of EC. The examination will then proceed towards the introduction of a collection of concepts important for the discussions of this thesis. Tool-assistant-agent triad will be used for classifying CAD systems in terms of the level and manner of their operational intelligence. The difference between online and offline agents will be described to point to the need for non-linear and open-ended problem handling and for dynamic agents. The Situated Function-Behavior-Structure (FBS) ontology of John Gero will be discussed at this point to illustrate the basic requirements of such agents.

The third main section of the chapter concerns an attempt to identify and map, first, the basic constituents of a design situation, and secondly, the difficulties encountered by the AI studies in design as linked to the corresponding strategies and techniques employed by human designers against these difficulties. Additionally, the multifarious evaluations required by design situations will be classified with a fourfold scheme. It appears from the preparatory examinations for this attempt that to carry out the integral and dynamic exploration characteristic of design, a design agent requires building design expertise over an interpretative and evaluative daily understanding.

Thus, by establishing a basis regarding the core characteristics of design processes, required capabilities of design agents will be revealed. This makes it possible to question, whether such technologies that would provide these capabilities currently exist or not. It becomes also possible to set forth a research program and to situate EC within this program. Thus, the remainder of the chapter focuses on a research program, whose ultimate aim would be total or partial, i.e., strong or weak, Artificial

/ Autonomous / Automated Design (AD). In brief, it will be claimed that Strong-AD is AI-complete, in other words, the problem is within the group of the hardest problems of AI. Thus, design appears as a challenge for AI in general. Therefore, AD's basic assumption should be that current human strategies and behaviors in design are the proper responses to design situations. The second assumption will state that, design computation takes place in a design environment where the humans and CAD tools operate together. At least until general machine intelligence will be available, Computational Design studies will have to tackle their problems through human-tool complexes. While weak-AD is the practical paradigm for Computational Design, strong-AD may remain an ideal or an ultimate target. The only viable way towards this ultimate aim is the cooperative human-tool complexes that would mutually develop while working together. As will be discussed with reference to Nicholas Negroponte's research program for "The Architecture Machine", the basic requirements for this aim are, first, the employment of machine-learning technologies, and secondly, voluntary acceptance of artificial agents for cooperation.

The overall task that appears after all these examinations is to develop intelligent artificial assistants that would be able to operate together with the human agents while learning from them. This is also valid for EC based systems. The importance of this plain proposal is the displacement of the emphasis from rigor or technical success to usability, which would require ease-of-use and pleasure no less than utility, scientific rigor, or technical competence. The response for these requirements will be an integrated approach, i.e., the "design_proxy" approach, for guiding the development of artificial design assistants. The last sections of the chapter will be introducing this approach with its main tenets.

2.1 Design as Ability, Activity, and Profession

The term 'design' is the common name of a series of professional fields. It denotes both the core activities of the professionals that are active in these fields and the products of these activities. While it is sometimes convenient to talk about design and designers as if these terms denote a homogeneous field, there is no single unified discipline as 'Design'. Instead, there is a whole range of designers practicing within more or less distinct fields, such as software design, interaction design, graphic design, textile design, fashion design, product design, stage design, interior design,

landscape design, structural design, architectural design, urban design, etc. Each of these fields may contain further ramifications. Architectural design is a prime example of this ramification with a multitude of sub-fields corresponding to different sub-problems such as facade, structure, acoustics, air conditioning, illumination, etc.

Design professions are diverse in terms of their productions, organizations, and methods. Nevertheless, these professions resemble each other in some essential ways and they share common practices. Thus, to some extent, we can see design as one broad family, which comprises different design professions and activities (Dorst, 2006, p. 34; Lawson, 2005, p. 4).

Contrary to this professions-based definition, design has also been seen as a fundamental everyday activity that all humans perform. It is a natural human ability and all people are designers to a greater or lesser extent. For instance, people design their appearances throughout their lives. This is a complicated act, since people dress for multiple aims. It requires integrated strategies in order to deal not only with unstable weather conditions, but also with looking nice and with communicating adequate messages about who the person is and how she regards others. On a different level, planning and organizing someone's daily activities can be seen as a kind of design activity, too. Likewise, designing, arranging, and personalizing living and working spaces is an important part of daily human life.

Just like the process of language acquisition, these kinds of skills tend to be developed in a continuous process, which begins in early childhood (Lawson, 2005, p. 4, 234, 236; Cross, 2006, p. 100). However, while a kind of basic design ability is being shared by all humans, practicing designers exhibit differences in some respects. To be able to produce beautiful, practically useful, and well-functioning products, they need to develop subtle and multifarious skills including both generic design abilities and expertise in their specific field (Lawson, 2005, p. 32, 234). Additionally, they require a multifaceted intellectual knowledge base in a wide variety of issues that interest societies. While developing this heterogeneous and mostly tacit kind of knowledge, designers become experts in their fields. This requirement for the gradual development of a kind of design expertise indicates that design is not a mystical ability but a combination of skills whose basis is provided by basic human abilities. In any case, in order to tackle complicated problems, design ability must be further gained and developed over this basis.

In a pragmatic sense, one essential activity of the designers is to provide design proposals in the form of detailed production descriptions for an artifact (Cross, 2006, p. 15-16). These descriptions usually specify materials and geometric properties and they are usually, but not exclusively, delivered in the form of drawings. From the viewpoint of design, the crucial constituent of this process is neither the drawing, nor the delivery process, but the process of the generation of design proposals; from a set of initial needs, requirements, and intentions to a new bit of reality, consisting of a—most probably physical—structure and an intended use. This constitutes the fundamental activity of the designers (Cross, 2006, p. 16; Dorst, 2006).

For such aims, design requires resolving vague situations, adopting solution-focusing strategies, problem-finding as well as problem-solving, analysis as well as synthesis. It requires employing inductive, deductive, and productive thinking and using verbal, non-verbal, graphic, and spatial media both to carry out the thinking process and to communicate ideas. Moreover, designers have to make judgments and balanced decisions often in ethical contexts (Lawson, 2005, p. 14, 233; Cross, 2006, p. 19; Rowe, 1987, p. 102).

2.1.1 Design research

In its explanations and proposals, this thesis relies heavily on the ideas, models, and explanations borrowed from current design research. Therefore, a brief description about the state of design research will be presented in this section.

Design research refers to the body of work, which attempts to improve our understanding of design through systematic and reliable methods of investigation (Cross, 2006, p. 99; Bayazit, 2004). For the development, articulation, and communication of the knowledge related to design, design research draws on three main sources. These are people, processes, and products (Cross, 2006, p. 100). Thus, the objectives of design research are, to study, research, and investigate the entities (products) that are made by human beings (people), and the ways these activities have been carried out (process); both in academic studies and in professional practices (Bayazit, 2004).

Design research mainly studies (Cross, 2006, p. 101; Bayazit, 2004):

1. Designerly ways of knowing and the practices and processes of design, i.e., how designers work, how they think, what and how they know, and how they carry out and manage design activity.
2. What is achieved at the end of a purposeful design activity, i.e., form, configuration, and physical embodiment of man-made things, how these things perform their jobs, how they work, how they appear, and what they mean.

This thesis is explicitly concerned with the first area. In an attempt to assess where and how Evolutionary Computation can be used within design processes, the study undertakes an investigation of how human designers carry out design. On the other hand, because building proposals are integral parts of architectural design processes, the second area will also be visited implicitly. Firstly, as part of the idea of Architectural Stem Cells (see Chapter 4), in an effort to define suitable tasks for evolutionary computation, and secondly, in terms of the methods that will be introduced to analyze existing building layouts, for the application of evolutionary techniques to architectural design.

Although design involves some skills that are generic, in the sense that they apply to all forms of design practice, it also seems likely that some skills are specific to certain design fields (Lawson, 2005, p. 32). Beyond the basic commonalities, this thesis will be skewed towards issues related to the group of design fields that seems to lie near the middle of the spectrum of design activities, i.e., architecture, interior design, product design, and graphic design, and may omit some issues specific to more systematic fields of design, i.e., engineering design.

There is a shared confidence in the relative maturity of the current state of design research and theory compared to its early times, and it can be claimed that, after several decades of research, a firm corpus of design theory has been collected, which can reliably guide further research, interpretation, and education.

First of all, the general outline of the design realm has been delineated, and there is a series of ideas that appear to be held in common regarding design processes, design education, and design professions. In particular, several core characteristics of a design situation, i.e., the co-evolution of the problem-solution pair, contextual

character of design, reflective conversation, and an essential variability of the strategies of different designers have found detailed expressions within design theory. In addition, although in rather broad outlines, there are some common assumptions about which approaches could work while designing and which would not.

However, it should be stressed that explicit procedural knowledge on 'how to design' is rather sparse and loose, if not altogether missing. The first difficulty in the endeavor to explicate designerly knowledge is the tacit nature of most design knowledge and skills (Niedderer, 2007). Design requires an extensive amount of training and expertise and this clearly indicates the presence of a designerly kind of knowledge. However, most of the times, this knowledge is not expressed in a propositional form. Human designers know how to design in a tacit way, but the endeavors to explicate their experiences and procedures have not brought about a body of knowledge, or a textbook, that could help designers learn and develop their skills better, or produce better designs. A quest for a 'scientific design' is mostly abandoned and this is related to the situated and highly context-dependent nature of design activity. It does not seem likely that there will ever be a general method for design, because each design situation is extremely complicated and unique. Thus, practitioners speak less of clearly predefined routes but rather more of their own individual interests, personal approaches, and strategies. The writings of practitioners confirm the view that there is not one route through the design process, but many (Lawson, 2005, p.183).

A consequence is, as design is being better understood, we are moving farther away from describing specific methods for design. We do not—and most probably will not—have simple, common laws on how to design in a particular design situation. Instead, within design theory, a mixture of ideas is being gathered, containing on the one hand, broad outlines for general directions to be taken while designing, and on the other, stories of particular strategies of specific designers. There is no straightforward way for researches to proceed towards prescriptive models, or to convert design theory into practical proposals for designing better.

Another consequence is, despite many decades of research in Computational Design and despite considerable developments in some other fields related to artificial

intelligence, for many design problems, automated approaches are still not available: design remains a human endeavor.

Obviously, these conclusions are not in line with Herbert Simon's (1996) proposal for a science of design. Simon claimed that the prerequisite to introducing artificial intelligence into the design process was an explicit and precise design theory. He also thought this to be the key to establishing design theory's academic acceptability and its appropriateness for a university system (Simon, 1996, p. 116).

Contrary to Simon's expectations, design theorists adopted alternative routes that divert the field from a quest for a precise, scientific design. This was not because explicit knowledge would not help Computational Design studies, but that this kind of knowledge did not exist for design. With regard to their knowledge structures, design fields are different from both natural sciences and engineering fields.

As the field grows more mature and its participants become more confident about their subject, a better awareness is developed on the issues related to the methodology of design research. Since its inception, research on design processes has utilized a multiplicity of empirical methodologies, which commenced with introspection and assertion of a priori models, and moved forward to include carefully controlled laboratory experiments (which are mostly inspired by psychology and natural sciences) and protocol studies on constrained laboratory cases (inspired by cognitive science) (Lawson, 2005, p. 258).

These empirical methods yielded an amount of information for a better understanding of design processes. However, recognition of the inadequacies of laboratory-based empirical methodologies in the face of the complexity of design situations put forward the role of methods like interviews and long-term observations, which lean more towards the qualitative side and require superior interpretative skills on behalf of the researcher. This recognition gave way to observations within more realistic conditions in preferably design offices, and more recently, to interviews and long-term investigations of real practices (Lawson, 2005, p. 258; also see, Cross, 2011). In a connected line of thinking, it has been suggested that researchers who at the same time practice design usually have a better understanding of the complications of this activity (Lawson, 2005). This claim is

strongly related with the supposition regarding the tacit expertise possessed by the designers.

Considering the above mentioned complexity of the field, and the special tacit knowledge possessed by the researchers coming from design related education and practices, a claim for a separate disciplinary status for design research can be raised:

> What has been happening in the field of design research is that there has been a growing awareness of the intrinsic strengths and appropriateness of design thinking within its own context, of the validity of a form of 'design intelligence'. There has been a growing acceptance of design on its own terms, a growing acknowledgement and articulation of design as a discipline. We have come to realize that we do not have to turn design into an imitation of science; neither do we have to treat design as a mysterious, ineffable art. We recognize that design has its own distinct intellectual culture (Cross, 2006, p. 101).

During its brief history, design research has always proceeded side by side with Computational Design. Throughout the first generation of design research (so called, the "Design Methods Movement"), attempts at finding better and systematic ways for design culminated in several models of design processes. Informed by these models, the newly established Computational Design field attempted to produce practical approaches to aid and carry out design. The models produced for understanding design served for practical applications and trials (computational or not) and results from these applications (and most importantly the interpretation effort on these mostly failing experimentations) helped inform design research about the shortcomings of the models; hence a better understanding has been gradually developed.

Therefore, for both practical and theoretical reasons, proposing tentative models has been an important method of design research. At any time, there should always be a multitude of these models, just because of the complicated and multifaceted nature of the field. These models help produce layers of understanding towards different and sometimes incompatible aspects, targets, and scales of design. This is one of the reasons why researchers in the field of Computational Design should continue proposing new models and experimenting with their consequences in practical situations. An understanding of human designers' strategies will help develop a better understanding in terms of the limitations and shortcomings of computational approaches, and devising new models, approaches, and methods is the only viable way to move forward in the current situation of the field. This also explains the

methodology adopted for this thesis. The thesis cites some existing models in detail as well as offering both theoretical and practical models itself, targeting different layers and levels of the design realm.

2.1.2 Design as purposeful transformation and planning: the problem-solving paradigm

Design can be characterized as an activity of transformation, where the transformation is intended to have some desired results, although these results are broad and cannot be precisely defined before the process is experienced.

John C. Jones (1970) proposed what he regarded as the "ultimate definition" of design as "to initiate change in man-made things". Similarly, according to Simon (1996, p. 111), “everyone designs who devise courses of action aimed at changing existing situations into preferred ones". For Simon, the intellectual activity that produces material artifacts is not fundamentally different from the one that prescribes remedies for a sick patient or the one that devises a new sales plan for a company or a social welfare policy for a state (Simon, 1996, p. 130). In a similar sense, design is sometimes related to planning. Designers plan the behavior of the design and its users, as well as production of the design (Dorst, 2006, p. 32).

These interrelated definitions are closely linked to the definitions of design brought forward by the problem-solving paradigm. Problem-solving requires that the problem solver recognizes a state of affairs that needs improving and a target state of affairs that would represent the improvement, where the ways to get from the unsatisfactory state to the improved state are not readily apparent. In this rather general sense, design can be seen as a kind of problem-solving.

The problem-solving paradigm has its origins in the field of mathematics (Gedenryd, 1998, Chapter 1). It may be described as a collection of assumptions, approaches, models, and methods, and has been first adopted for AI studies by Herbert Simon and Allen Newell (Newell and Simon, 1972). The methods that were used by Newell, Simon, and Shaw during 1950's, in one of the seminal studies of artificial intelligence—with the aim of developing a computer program that could prove logic theorems—eventually resulted not only in a practical automated problem-solving approach (General Problem Solver, GPS) but also in a general theory of human problem-solving (Newell and Simon, 1972). This paradigm, in which design is seen

as a rational problem-solving process, has been a dominant influence shaping prescriptive and descriptive design methodology and a considerable portion of the work done in design methodology has followed it in its assumptions, view of science, goals, and methods (Dorst and Dijkhuis, 1995; Gedenryd, 1998).

One of the reasons for the success of this paradigm was that it was proposing a much-needed rigorous basis for design methodology. Problem-solving theories introduced by Simon and Newell provided a framework for design research by allowing the study of designers and design problems within a paradigm of technical rationality; in the logic-positivistic framework of natural sciences, taking sciences like physics as the model for a science of design (Dorst and Dijkhuis, 1995). A systematic science of design would also be convenient for practical reasons. This kind of knowledge, if available, would render design amenable to computational techniques, and would show the easy way towards design automation.

In this approach, there is much stress on the rigor of the analysis of design processes, objective observation, and direct generalizability of the findings. Logical analysis and contemplation of design are the main ways of producing knowledge about the design process (Dorst and Dijkhuis, 1995). However, the stress on objectivity, rigor, and generalizability poses problems when applied to design fields. In contrast to well-defined (alternatively: well-formulated, well-structured) problems, the problems that the designers tackle often exhibit characteristics that are referred to as ill-defined, ill-structured (Simon, 1973), open-ended and even as wicked (Rittel and Webber, 1973).

This contrast is important, because as we move away from well-defined domains like chess and puzzle solving, and get close to more open-ended domains like planning and design, cognitive science's ability to explain the relevant cognitive processes considerably diminishes. This is mainly due to an increase in the complexity of the problems. In addition, usually the increase in complexity cannot be compensated by a linear increase in the problem-solving effort. Because, apparently, as the problem becomes more complex, the problem-solving behavior changes in a qualitative sense, and altogether different problem-solving attitudes become necessary (Lawson, 2004, p. 19).

Indeed, Simon (1996) well acknowledged the ill-defined character of design problems, but he still went on to argue in favor of a systematic design science. In his view, ill-definedness had to be overcome, rather than being accepted as the regular situation to work with. One of the problems that he considered was a requirement for special reasoning mechanisms pertaining to the prescriptive character of design, which poses problems regarding the use of formal logic. After describing several attempts to utilize different types of logic, Simon pointed to the area of design practice where, for him, standards of rigor were sufficiently high, i.e., optimization methods. According to Simon (1996), once formalized, optimization problem was a standard mathematical problem to maximize a function that is subject to constraints. Thus, the answers could be reached in a deductive fashion, using the standard logic of the predicate calculus. Therefore, Simon proposed optimization theory as a prime example of what he believed a science of design could and should be (Simon, 1996, p. 117, 118).

A scientific design that focuses on optimization techniques can only be applied to well-defined problems already extracted from situations of practice. However, as will be discussed in the following sections, in most design situations, well-defined problems are not given but must be constructed from messy problematic situations (Schön, 1983, p. 47), and the primary task is not the optimization procedure, but the definition of the problem itself.

It should be noted that, Simon (1996) did not limit the subject of computational techniques to optimization. He pointed to the traditional engineering design methods that make practical use of satisfactory performance, rather than insisting on optimal results. This is why, he introduced the term "satisficing" (Simon, 1992). Where optimization is not possible—and he admits that this is the way problems usually pose themselves in actual design situations—he suggested decision making methods that compare designs in terms of "better" and "worse", rather than "best" (Simon, 1996, p. 119).

In practice, the problem-solving approach mostly means structuring a solution process as a "search" procedure for better solutions amongst a limited set (Dorst and Dijkhuis, 1995). The scope of the steps taken towards a solution is limited by the information processing capacity of the acting subject. This limitation of the reasoning capability is referred to as "bounded rationality" by Simon (1992).

Indeed, the 'pose-search-evaluate-choose' process can be recognized in real design processes. This is why, seeing design as problem-solving does capture some aspects of design and the idea that design is problem-solving has led to the development of phase models of the design process; i.e., analysis-synthesis-evaluation models (Lawson and Dorst, 2009, p. 30-31), which will be discussed in the following sections. In the problem-solving paradigm, the design process can be said to comprise three major tasks, which in a way echoes the tripartite phase models of design process (Kalay, 1992):

1. Defining a set of desired conditions that comprise the objectives to be achieved: Analysis (goals, objectives, performance criteria, constraints, etc.)
2. Specifying actions that will achieve the desired objectives: Synthesis / Generation (operators, modifiers, etc.)
3. Predicting and evaluating the effects of the specified actions to verify that they are consistent with each other and they achieve the desired objectives: Evaluation (simulation, testing etc.)

From a practical viewpoint, it can be suggested that as long as the design goals are explicit, clear, and stable and when it is possible to generate a set of comparable solutions, a design problem can be tackled through problem-solving approaches. This occurs more often in the technically oriented design professions, like engineering design, and also in the latter phases of a design project, when most of the conceptual decisions have already been taken (Lawson and Dorst 2009, p. 30, 31). Dorst (2006) refers to these essentially well-defined problems as "closed problems". When such a solid ground is encountered in a design project, the problem-solving model and its accompanying methods can help in structuring the design process (Dorst, 2006, p. 15, 16).

However, it would not be wise to think that all design activity can be captured within the problem-solving model (Lawson, 2005, p. 31). Not all design processes are exclusively composed of closed problems. There are also open problems within the design realm, which are widely referred to as ill-defined problems. In reality, every design situation involves both open and closed types of problems. The intensity and essentiality of the open problems in a design situation would render its character as open or closed.

Closed design problems are more like puzzles. Solutions are tried out, and the feedback from evaluating the solutions is immediate and clear. On the other hand, open problems require the designer to play with concepts and ideas through a wide range of possibilities before settling on a firm direction. Ideas are proposed, critically inspected, and continuously reconsidered. This is done repeatedly, making gradual improvements as the designer learns more about the problem. Therefore, design involves finding appropriate problems, as well as solving them, and includes substantial activity in problem structuring and formulating, rather than merely accepting the problem as given. These two types of problems require different mental strategies (Dorst, 2006, p. 15, 16, 35; Cross, 2006, p. 77).

A recent cognitive neuroimaging experiment by Alexiou, Zamenopoulos, and Johnson (2009) supports the idea that these two types of problems, i.e., open / ill-defined and close / well-defined types—which were equated to design and problem-solving respectively—require different mental strategies. The study involves functional Magnetic Resonance Imaging (fMRI) of volunteers while performing design and problem-solving tasks.

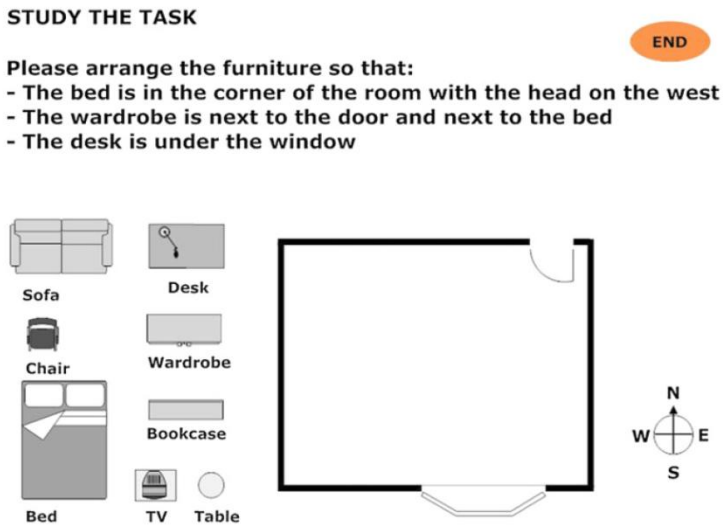


Figure 2.1 : Problem-solving task used in the experimentation by Alexiou, Zamenopoulos and Johnson (2009).

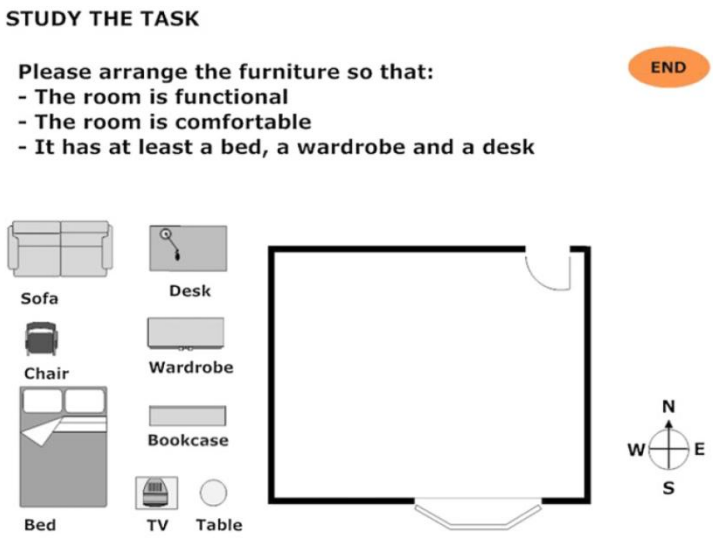


Figure 2.2 : Design task used in the experimentation by Alexiou, Zamenopoulos, and Johnson (2009).

In the problem-solving task, a criterion for deciding the termination of the task is given, as well as a definition of legal moves. The problem itself is well-defined, the legal moves are known and there is a unique set of equivalent solutions (Figure 2.1). In contrast, in the design task, there is no predetermined final state or criterion for deciding the termination of the task. The task is open-ended, and requires defining the problem as well as the solution space. It requires the creation and interpretation of a set of moves, as well as the creation of criteria for evaluating the solutions (Figure 2.2).

The findings of the study suggest that design and problem-solving behaviors involve distinct cognitive functions associated with distinct brain networks. There is a more extensive neural network involved in the activity of understanding and resolving design tasks than the network involved in problem-solving tasks (Alexiou, Zamenopoulos, and Johnson, 2009).

Although it is possible, in a broader sense, to name all of these activities as types of problem-solving, the specific connotations of the phrase problem-solving renders this term inappropriate for denoting most design activities. Although somewhat reductive, this thesis will follow the same dichotomy that is proposed in the above-mentioned study, where problem-solving and design correspond to the two ends of a scale of solution strategies. Throughout this thesis, the term design will mainly be used to indicate this open type of problem-solving, in contrast to the closed type, which will be referred to as problem-solving.

The kind of problem-solving in design essentially deals with rather open-ended and ill-defined problems and this is the reason to differentiate it from other forms of problem-solving that are characterized by well-defined problems. In the problem-solving approach, the problem definition is supposed to be stable; however, this is rarely the case in design. Moreover, in problem-solving, a solution space has to be defined before the solution process starts, while, in the form that they are carried out by human designers, design problems are rarely understood before the solution process starts.

For closely related reasons, at least until its detailing stages, a design process cannot be identical to optimization. The optimization approach assumes that the essential needs of a circumstance can be listed and can be expressed in a measurable form before the solution process starts. For design problems, even if this was possible, the quantities of criteria, constraints, or needs could easily reach huge numbers[2]. Moreover, these requirements mostly reside on incommensurate levels, so that comparing them is also a problem itself. After all, designers are rarely completely sure of these needs, let alone being able to formulate them in a measurable form.

Design does not have a predetermined domain. In fact, design often begins without any clear statement of the problem as a whole. Some general objectives may exist, but there is rarely an unambiguous way of knowing how well one is doing as one proceeds. Even more confusingly, it might not be possible to arrive at an overall assessment to compare relative values of various solutions. Design solutions are not certainly right or wrong, contrary to most puzzles, and there are no knowable optima (Dorst and Dijkhuis, 1995; Lawson, 2004, p. 20). Considering this issue, Lawson presents a compelling comparison of design with chess, which is taken as an example of the well-defined problems:

> Designing then, in terms of chess, is rather like playing with a board that has no divisions into cells, has pieces that can be invented and redefined as the game proceeds and rules that can change their effects as moves are made. Even the object of the game is not defined at the outset and may change as the game wears on (Lawson, 2004, p. 20).

[2] For an illustration of this issue, see Christopher Alexander, "Notes on the synthesis of form", 1964. And for a concise criticism of Alexander's early approach, which was later abandoned by himself, see Lawson, 2005, pp. 75-76.

It should be noted that, an understanding of design in terms of problem-solving behavior and methodology deserves a more in-depth discussion, not only because of its great historical significance but also for its current importance in applied computational fields. In particular, Evolutionary Computation in design has largely followed this paradigm. Therefore, it is crucial for this thesis to elaborate and re-map the relationships of conceptions and approaches within this multi-level discussion. This investigation will be carried out in Section 2.2 under the general topic of Computational Design. Current section has been devoted to a general characterization of design and will proceed through complementary descriptions.

2.1.3 Design is solution focused: co-evolution and bridging

Designers do not treat design assignments as given. Instead, they interpret their assignments with an awareness of their own design environments, resources, and capabilities, which, when brought together, constitute unique design situations. The studies on design strongly support the idea that, within these complicated situations, architects, engineers, and other designers adopt a strategy, which is based on generating and testing potential solutions, rather than focusing on the problems. This strategy can be recognized in all design professions. Indeed, in many design problems, the generation of possible solutions and their gradual improvement is the only way forward (Cross, 2006, p. 18; Dorst and Cross, 2001; Dorst, 2006, p. 14).

This feature of design behavior arises from the nature of the design problems, in which not all the necessary and sufficient information can be provided at the outset. As a result, designers are accustomed to problems that do not easily lend themselves to prior analysis. In a design situation, some of the relevant information can only be found by generating and evaluating high-scoring solutions. The initial, conjectured solutions proposed by the designers are devices to actively analyze and gain understanding about a design situation (Cross, 2006, p. 18; Dorst, 2006, p. 14).

Concerning the nature of the information within design problems, Dorst (2004) identifies a threefold situation:

- First, design problems are partly "determined" by hard, unalterable needs, requirements, and intentions. A designer will have to reserve time in the early part of her design process to unearth these hard facts by information gathering and analysis. This type of information can be seen as a rather clear

input at the start of a design process and is compatible with the rational problem-solving paradigm.

- Secondly, a major part of the design problem is "underdetermined", in the sense that the interpretation of the design problem and the creation and selection of possible suitable solutions can only be decided during the design process and on the basis of the proposals put forward by the designer. A description of the design problem in terms of needs, requirements, and intentions can never be complete. Moreover, there is a rift between the solutions and the conceptual world of needs, requirements, and intentions, which means, neither the best solutions to a problem, nor the exact course of actions to take to reach the best, or even plausible solutions are known in advance (Lawson, 2005, p. 271, 272).
- Finally, part of the design problem can be considered "undetermined", in the sense that the designer is largely free to design according to her own taste, style, and abilities. Although the designer would still have to defend these aspects of her design to others, she is dominant in these areas, in the sense that it is she that will provide the criteria over which the design is to be judged.

The last two characteristics of the design situation oblige the designer to introduce her own strategies and a kind of information that has to be created or brought in by her. Because the problems are not predetermined, usually some extra structure and information, in other words extra ingredients have to be provided by the designer. Through her interviews with architects, Jane Darke (1979) observed how these extra ingredients imposed a limited set of objectives or a specific solution concept as a "primary generator". This strategy, which is utilized early on in the design process, enables the designer to structure and define a solution space, which is otherwise vague. This way the designer simplifies the variety of possible solution directions. Further understanding of the problem is gained by testing the solution proposals that are generated by following the early conjectures. Starting with this observation, Darke (1979) arrived in a tripartite model for this process in the form of generator (initial structuring idea), conjecture (proposed solution), and analysis (evaluation).

This multifaceted process can also be seen as an effort for matching or bridging between the problem – solution pair. As stated earlier, design is not a matter of first fixing the problem (through objective analysis) and then searching for a satisfactory solution concept (Dorst, 2004). It is more a matter of developing and refining together both the formulation of a problem and ideas for a solution. This involves a period of exploration in which problem and solution spaces are evolving in parallel and are unstable until temporarily fixed by an emergent bridge which identifies a problem – solution pairing. The designer is seeking to generate a matching problem – solution pair, through a co-evolution of the problem and the solution (Dorst and Cross, 2001; Dorst, 2004). Therefore, design problems and design solutions are inexorably interdependent. It is meaningless to study solutions without reference to problems (Lawson, 2005, p118).

It should also be noted while passing that design solutions are often holistic responses to design situations. Parts of design solutions often serve several purposes in an integrated manner. This corresponds to the highly complex and multi-dimensional character of the design problems. It is frequently necessary to devise integrated solutions to a set of requirements and good design is usually an integrated response to a whole series of issues (Lawson, 2005, p. 58, 59, 62). It is "rarely possible to dissect a design solution and map it onto the problem saying which piece of solution solves which piece of problem" (Lawson, 2005, p. 122).

This view can also be expressed as a negotiation between problem and solution, or between what is desired and what can be realized. Designers negotiate a reconciliation between the two views of the situation, which are problem and solution views. The problem view is expressed in the form of needs, desires, wishes, and requirements, while the solution view is expressed in terms of the physicality of materials, forms, systems, and components (Lawson, 2005, p. 271, 272).

This aspect of the design process can be described in terms of Maher, Poon, and Boulanger's (1996) model for the co-evolution of problem and solution spaces (Figure 2.3).

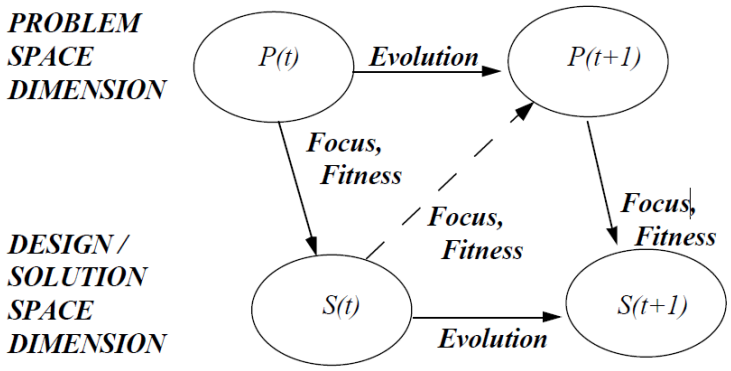


Figure 2.3 : The co-evolution model of Maher, Poon, and Boulanger (1996).

Maher, Poon, and Boulanger's model principally aims at practical Evolutionary Computation applications, as presented by their 1996 paper. However, by interpreting a protocol study, Dorst and Cross (2001) showed how a pattern of real design development could be modeled along the lines of this co-evolution model. In their particular example, first, a chunk of coherent information is formed by the designers using the assignment information (problem space is being structured), this in turn helps to crystallize a core solution idea (solution space is being structured). This core solution idea changes the designers' view of the problem, and this results in the designers' redefinition of the problem (problem space is revised). Checking whether this fits in with earlier solution ideas, the designers modify the tentative solution that they had (solution space revisited and solution refined).

2.1.4 Design as conversation with a situation: reflection-in-action

As a reaction specifically to the so-called technical rationality, which brought forward the problem-solving view of design, an alternative paradigm has been proposed by Donald Schön (1983), which describes design as a conversation with the materials of a situation, through a process that is called "reflection-in-action". Schön describes design as a "reflective conversation with the situation", where the problems are actively set or "framed" by designers, who take action to improve the perceived situation (Schön, 1983).

According to Schön (1983, p. 18), designers, just like the other professionals, encounter complexity, uncertainty, instability, uniqueness, and value conflict in their practices. Design is a prescriptive activity where there is no infallibly correct process. Because the solution is not the logical outcome of the problem, there is no

sequence of operations, which will guarantee a result (Lawson, 2005, p. 124). Moreover, design involves subjective judgment and designers often have to make decisions in ethical contexts (Lawson, 2005, p. 124, 125), which often require intricate and context-dependent interpretations. Since design problems defy comprehensive description, they offer an inexhaustible number of solutions and the design processes cannot have an identifiable end (Lawson, 2005, p. 123). Any design problem is unique; therefore, a core skill of a designer lies in determining how every single problem should be tackled (Schön, 1983, p. 78). Design processes tend to be complex and in any particular design situation:

> . . . There are more variables—kinds of possible moves, norms, and interrelationships of these—than can be represented in a finite model. Because of this complexity, the designer's moves tend, happily or unhappily, to produce consequences other than those intended. When this happens the designer may take account of the unintended changes he has made in the situation by forming new appreciations and understandings and by making new moves. He shapes the situation in accordance with his initial appreciation of it, the situation 'talks back', and he responds to the situation's back-talk. In a good process of design, this conversation with the situation is reflective. In answer to the situation's back-talk, the designer reflects-in-action on the construction of the problem, the strategies of action, or the model of the phenomena, which have been implicit in his moves (Schön, 1983, pp. 78-79).

The essential part of the reflection-in-action view is that designers are active in structuring the problem. Designers work by "framing" a problem in a certain way, making "moves" towards a solution and evaluate these moves on the criteria of coherence, accordance with the specifications, and the value of the solution (Dorst and Dijkhuis, 1995). Goldschmidt (2006) defines a design move as the smallest step, either complex or simple, that repositions the designer in her exploration. From a cognitive perspective, such a step's average duration is a few seconds (Goldschmidt, 2006). The concept of design move is important for Computational Design studies in two respects. First, it is often conceptualized as a building block of design processes within protocol analysis studies. Secondly, there is a potential correspondence between such a move and a computational operation. However, caution must be taken while advancing this kind of comparison, because, even the simplest human design moves often involve complex verbal and non-verbal thinking that calls for a large variety of human mental faculties, while most computational operations are results of relatively simple practical procedures with a smaller range of aspects,

which are mostly technical. Therefore, it can be claimed that there is a kind of complexity gap between human moves and existing computational operators.

Nevertheless, the concept of 'rationale', if understood as the summary of the thinking that guides a move, may help bridge this gap; or rather, help divide design moves into explicit and implicit types in order to match only the explicit moves with a computational operator. In analyzing design protocols, Goldschmidt (2006) has found that the numbers of the two main types of design moves, i.e., arguments of rationale and physical arguments are roughly equal within design processes; which supports the claim that "design exploration is a complex activity in which proposals are considered against their rationale" (Goldschmidt, 2006). This topic will be further discussed with regard to design exploration in the following sections.

The reflection-in-action paradigm is very much in line with the solution focused co-evolution approaches to design and has gained considerable support as an alternative to the problem-solving view. However, the relative weakness of the underlying theory makes it hard to draw any general conclusions from this description of design (Dorst and Dijkhuis, 1995). No specific structure for design problems is offered and there is no basis for judging the appropriateness of a certain frame employed in a particular situation. This kind of description of the design process is highly problem dependent and would result in studies that are very hard to compare (Dorst and Dijkhuis, 1995). However, these deficiencies may be claimed to stem from the very nature of the design process itself.

Referring to the above view of design as based on conversation and perception, Lawson (2005, p. 266) draws attention to the narrative character of this approach and details design conversations into several dimensions, in terms of whether they are between actors in a design situation, between designers and representations, or carried out reflectively in the minds of individual designers.

Design progresses at least partly through the conversations that take place between the actors of a design situation, which includes clients, designers, and legislators amongst others. Thus, we can see design as a process of negotiation. Starting with disparate positions about some common purpose, the parties “come into the negotiation taking different views and having different objectives but with a

willingness to try to reach some form of agreement that all parties can accept" (Lawson, 2005, p. 271).

We can see design as a conversation even when performed by an individual designer. A designer carries on a conversation with a drawing (Lawson, 2005, p. 266). In this view, design representations are part of the mental process of thinking about a design, and sometimes the act of creating representations is performed not to communicate with others, but mainly to pursue a line of thought. As the representations develop, they enable the designer to 'see' new possibilities or problems (Lawson, 2005, p. 266).

The first important step in the design conversation is identification, which is similar to what Schön (1983) called "naming". The significant elements are not just named, but their very character begins to be explored (Lawson, 2005, p. 269).

Another frequently employed form of negotiation, i.e., framing, is the process of selecting a particular view of the situation (Schön, 1983). Experienced designers are likely to have their own ways of framing situations that they have used before and which have proved helpful in the past (Lawson, 2005, p. 276).

Primary generators are an example of the act of framing, where a particular idea is brought forward in order to start an inquiry into the problem. However, framing is a recurrent action all through the design process and can also be used to turn the problem around, describing it in different ways possibly in such ways that what appears to be difficult becomes clearer or what appears to involve conflict can be seen to be harmonious (Schön, 1983; Lawson, 2005, p. 276, 277).

2.1.5 Design as inquiry into a unique problem situation

Conversation is a way of inquiring into a unique design situation and design can be considered a continuous inquiry into a problem situation. By trying out different ways of looking at the problem and experimenting with various solution directions, a designer gradually gathers knowledge about the nature of a design problem and the best routes to take towards a design solution. The designer proposes, experiments, and learns from the results, until arriving at a satisfactory result (Dorst, 2006, p. 16). Therefore, the inquiry into the situation is at the same time the very act of generating a solution. The designer is a character who performs understanding and producing

simultaneously. We should not imagine pure withdrawal; the human designer is always involved. In this regard, design can be seen as a process of learning:

> . . . you sketch an idea and then look at it with a critical eye. This fresh look often immediately shows you what must be changed in order to improve the design. So you change it, and then you again look critically at your work, etc. Design can be described as a process of going through many of these learning cycles until you have created a solution to the design problem. In this way you learn your way towards a design solution (Dorst, 2006, p. 16).

Designers make moves, and then reflect on the result of their moves, which are mostly, but not exclusively, in the form of non-verbal representations. This "seeing-moving-seeing" process is crucial for design to take place (Schön, 1983).

These remarks explain only one part of the importance of representation in design processes. Indeed, the conversational function of representations is no less important. Design involves multiple levels of conversational representations (like presentation, instruction, and consultation drawings) as well as representations to be used as devices to conduct mental actions (such as calculation drawings, experiential sketches, diagrams, and proposition drawings) (Lawson, 2004). The proposition drawing is the drawing where a designer makes a move, or proposes a possible design outcome. Therefore, it is “right at the very centre” (Lawson, 2005, p. 45), and is the kind that Schön had in mind when he wrote about the reflective conversation (Lawson, 2005, p. 46).

2.1.6 Phases of design and decomposition of design processes

There is an understandable progression of the design process from a raw, undetailed, and vague state of solutions towards a detailed specification of a product. This progression sometimes also involves the scale of the representation that is being considered; from a more general, overall scale towards details. Although this type of progression happens frequently, a significant number of highly proficient and reputable designers are reported to work to some extent from the detailed to the general (Lawson, 2004, p. 47).

Indeed, Lawson (2005, p. 48) considers the idea of the design process as a definite sequence of activities "rather unconvincing", as in the phase models of design which define design in terms of analysis-synthesis-evaluation cycles. Within this model, analysis is the ordering and structuring of the problem, synthesis involves attempts to

move forward and create responses to the problem, i.e., the generation of solutions, and evaluation denotes the appraisal of suggested solutions against the objectives identified in the analysis phase (Lawson, 2005, pp. 36-37).

It is reasonable to argue that, for design to take place, sometime during the process a brief should be assembled, the requirements should be studied and understood, one or more solutions should be produced and tested against some explicit or implicit criteria, and the design should be communicated to clients and constructors. Following a reasonably structured progression can lead to greater design success. However, according to Cross (2006, p. 92), ". . . rigid, over-structured approaches do not appear to be successful. The key seems to be flexibility of approach, which comes from a rather sophisticated understanding of process strategy and its control". Likewise, Lawson finds the idea that the above-mentioned activities occur in an order, or even that these are identifiable separate events very questionable, and he proposes a rather chaotic alternative (Figure 2.4):

> Our final attempt at a map of the design process shows this negotiation between problem and solution with each seen as a reflection of the other. The activities of analysis, synthesis and evaluation are certainly involved in this negotiation but the map does not indicate any starting and finishing points or the direction of flow from one activity to another. However, this map should not be read too literally since any visually understandable diagram is probably far too much of a simplification of what is clearly a highly complex mental process (Lawson, 2005, p. 49).

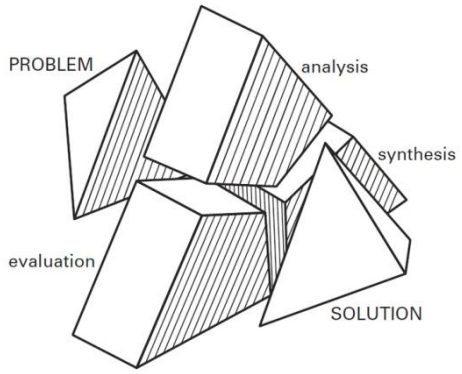


Figure 2.4 : Analysis – synthesis – evaluation model of Lawson (2005, p. 49).

Lawson's model (Figure 2.4), when understood as a critical endeavor, captures the difficulties that are encountered when trying to devise generalized models for design. Because of the large variety amongst observed design processes, the model is kept on an abstract, picture-like level and it risks an understanding as if there are no

discernible differentiations of actions or phases within the design processes at all, which is not the case as is shown by protocol analysis studies that document the micro variations within design processes[3]. Unfortunately, these studies are often limited to laboratory cases while the main phases within a complicated design process might span over several months.

Although the progression and main phases vary through each unique design process, these nevertheless take place, and as will be discussed in the following sections, this fact lies at the basis of a dynamic decomposition of complex design problems. For example, in architectural design, the required outputs progress through several scales and levels of detailing, from the initial briefing to the production drawings. Starting with a large and less detailed scale often helps studying the overall decisions, which have to be taken beforehand—at least tentatively—in order to proceed towards scales that are more detailed. This does not mean that a study over a specific detail could not be taking place simultaneously. The sequence and layering of phases is highly context-dependent and most of the times this is also a decision of the designers. In some contexts, especially in a team design effort and in complex large-scale building projects, a more orderly progression could work better. However, when it comes to product design, or a small-scale spatial intervention, the detail might become the generator of the whole project and overall decisions might follow later.

For these reasons, it might be better to think about the subdivisions of a design process in terms of task layers, scales, or viewpoints, rather than temporal phases. All the layers and scales have different characteristics, and each viewpoint yields a different result, focusing on different aspects of a design situation.

2.1.7 Models for the design realm

Bryan Lawson and Kees Dorst (Lawson, 2005; Lawson and Dorst, 2009) have proposed several models for design, each of which approaches design from a different viewpoint and with different aims. Taken together they can be claimed to draw a consistent outline for design. Three of these models will be presented here to conclude the account of design that is adopted for this thesis.

---

[3] For collections of design protocol analysis studies, see, Cross, Christiaans, and Dorst (1996) and McDonnell and Lloyd (2009).

The first model proposes a way to characterize design problems in terms of types, flexibility, and sources of problem constraints (Figure 2.5). Constraints in design result from required or desired relationships between various elements and determine the issues, which must be taken into account when forming the solution. The purpose of constraints is to ensure that the designed system or object performs the functions demanded from it as adequately as possible. Constraints function in a restrictive manner and they narrow down the designers' space of alternatives. The term is also often used to denote evaluative criteria, which might be more flexible (Lawson, 2005).

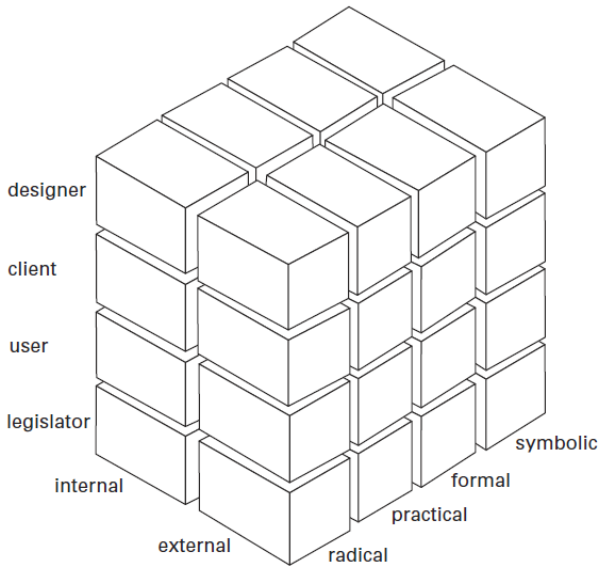


Figure 2.5 : The model of design problems via constraints (Lawson, 2005, p. 106).

In the model depicted in Figure 2.5, the first dimension, 'internal', denotes the constraints that establish relationships between elements of the object being designed, while 'external' constraints establish a relationship between some element of the object and what already exists. For example, in a housing design problem, an example of an internal constraint might be the proportions of a specific room, while the external constraints relate the designed object to its context, i.e., to the site boundary, the sun, the street, etc. that which are independently present. The definition of such problems is also a matter of deciding how much of already existing elements can be called into question (Lawson, 2005, p. 92, 93).

The second dimension concerns the types of the constraints. The 'radical' constraints are those, which deal with the primary purpose of the object or system being

designed. Radical is used here in the sense of fundamental. These constraints are the main reason for having the design in the first place. Such constraints are also often so specific and local to the problem that they rarely offer an opportunity for more generic investigation (Lawson, 2005, p. 103).

The 'practical' constraints deal with the reality of producing, making or building the design. They offer fertile ground for guiding principles especially for those designers who are fascinated by the materiality and process of making things (Lawson, 2005, p. 103, 170). The 'formal' constraints are those to do with the visual organization of the object. They may include rules about proportion, form, color, and texture (Lawson, 2005, p. 104). Finally, the 'symbolic' constraints concern the expressive qualities of design and the use of form and space to achieve specific effects rather than as an abstract assembly (Lawson, 2005, p. 105).

On the third dimension, the sources or generators of the constraints are situated. The client is the obvious example of a source of design problems and constraints, and is responsible for the majority of the radical constraints as well as for some of the symbolic ones. However, it would be misleading to think that a client presents a designer with a complete brief in which the problem is totally defined and the constraints are clearly articulated (Lawson, 2005, p. 85, 105).

The designer is the main generator of the formal and practical constraints, while also contributing symbolic ones (Lawson, 2005, p. 105). The range of possibilities can be restricted by initially focusing attention on a limited selection of constraints and moving quickly towards some ideas about the solution. In essence, this is related to the idea of the primary generator. Moreover, designers develop their own sets of guiding principles and these often determine the direction for a design project (Lawson, 2005, p. 188, 189).

Users are generally more remote from designers than clients are. In some cases, there may be no formal access to the users at all. Although often not involved in the actual design itself, legislators create constraints within which designers must operate (Lawson, 2005, p. 87, 89).

There is a final dimension of this model regarding the flexibility of the constraints. Each of the generators of design problems imposes constraints upon the design solution, but with different degrees of rigidity; the most rigid being those imposed by

legislators and the most flexible being those generated by the designer. According to their types, radical constraints are on the rigid end, while the symbolic constraints are on the flexible. Likewise, internal constraints are more flexible than the external ones.

The second model (Figure 2.6) describes design process through groups of activities and skills that are commonly observed in successful design. Main activities include formulating, representing, moving, evaluating, and managing (Lawson, 2005; Lawson and Dorst, 2009, p. 51). As such, this model can be seen as a guide for determining types of design moves and corresponding types of operators.

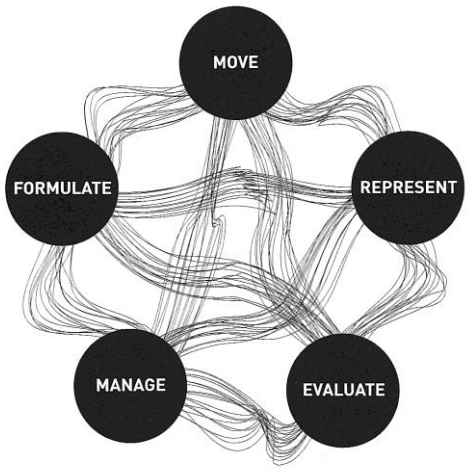


Figure 2.6 : Model of design activities (Lawson and Dorst, 2009, p. 51).

'Formulating' involves activities of identifying and framing. Designers must be skilled in exploring, finding, stating, and understanding problems, not only at the beginning of a project, but as a recurring activity. 'Representing' is also an activity that is continuous throughout the design process. It functions as an external memory, and helps designers inspect their own ideas, as well as communicating them. 'Moving' is inducing purposeful change in the design situation, such as creating new solution ideas and frames, or revising or refining solutions; hence, there are interpretative (rationale) and developmental (physical) moves. 'Evaluating' is divided into objective and subjective types. Design involves making judgments between alternatives along many dimensions that cannot be expressed on a common metric. Designers must be able to perform both objective and subjective evaluations and should be able to make judgments about the relative benefits of alternatives even when they rely on incompatible methods of measurement. Finally, 'managing'

comprises the higher level activities through which the process is monitored and guided (Lawson and Dorst, 2009).

The third model concerns the levels on which the above activities take place (Figure 2.7). These levels are 'project', 'process', 'practice', and 'profession'. Project level concerns a particular design problem, while process is a more general collection of methods or ways of working of a designer.

Figure 2.7 : Levels of design activities (Lawson and Dorst, 2009, p. 61).

Through their reflective moments, designers learn from their projects and develop their own approaches to design problems. This is crucial to developing a strategic view of design and possibly a distinctive style of designing. This brings us into the practice level, which comprises the style, assumed roles, participation, and partnership issues for a designer's professional practice.

2.2 Illustration by story-telling: the library problem

Until this point through the thesis, characterizations regarding design are kept in a rather abstract manner, in the sense that they did not concern concrete situations. With an aim to weave the aforementioned ideas and orientations together, the discussions have been kept on a theoretical level rather than a practical one. However, because design knowledge is mostly tacit, the design situation is best understood through design cases. Although each case is unique and has idiosyncratic aspects, many traits appear to be common among design situations. Thus, a description of a case, albeit imaginary, can help reveal the kind of knowledge that seems resistant to formulation into propositions or rules.

For this reason, following sections will illustrate the already mentioned points by way of an imaginary story of an architectural design problem. The imaginary case concerns a design assignment for a small district library in a Turkish town. The plot

appears complicated enough to illustrate the complexities of real design situations. In the following sections, why and how architectural design is complicated will be illustrated by means of this story. While the story unfolds, several issues regarding the complexity of architectural problems will be illustrated. The different issues that will be considered are not totally independent, thus a degree of overlapping is inevitable.

It should be noted that this narrative-based illustration is not proposed as a method to produce empirical knowledge. The following narration only aims at illustrating the issues that have already been addressed until this point with reference to established design research; this time in a more detailed and concrete manner. It should be seen like a mapping or a diagram, albeit a narrated one.

2.2.1 The problem: design problems tend to be vague

The imaginary problem concerns a design situation, which involves the design of a small-scale public library. A municipality in a medium-sized town decides to build a library over a building lot at an important crossroads within the town. The site was bestowed to the municipality by its owners, to be used for the benefit of the public.

At the time, when the library idea was decided, the municipality had already been using a part of the area as a recreational park. They also had intentions to use another portion of the land to generate some revenue. On the other hand, the donors of the field are against this kind of commercial use. The municipality claims to have the legal right to decide on the issue, yet, it is not clear how the law mechanism would decide on the issue.

In post-facto descriptions of design situations, most of these issues are simply omitted, especially if they appeared inessential in the end. However, in reality, the problems usually unfold in a gradual manner, and it is not possible to foresee which of the preliminary issues will come off as important in the end as well.

The first issue that will be identified here is the underdetermined or undetermined characteristics of a design situation. The problem is too vague in this stage; in other words, there are not many determined issues. The only rather well-defined sub-problem concerns cost estimation. Drawing from earlier projects and standard fares, overall budget may be planned. Yet this rigorous operation would only yield estimates within a very broad range. Although there would be estimates from

standard fares, at this stage the budget will not be exact, because first, it will depend on the proposal itself, and second, instabilities of the economical parameters like inflation rate and costs of materials and services will tend to keep the financial status in fluctuation. In addition, budget will depend on the type of contract; each type of contract might bring forward different behaviors from the sides. During the production phases, involvement of contractors and subcontractors for different aspects of the process will further complicate the issues. Even with the building of a small library, the money and effort that are mobilized are considerable, the decisions are important, and each involved party might like to have a say on the basic decisions. However, the partners and participants of the problem are not determined in the first stage, and their relative positions are usually dynamic and subject to negotiation.

2.2.2 Moving on: framing, proposing, negotiating

The design situation comes out in rather vague terms. There are a few basic issues that loosely delimit the problem. Even when the problem is strictly briefed, designers might tend to understand it as if it was not (Cross, 2006). The first thing that our architect has to do is to make the situation a bit better defined, preferably together with the client side. However, in this case the client side comprises several parties, the municipality, the ministry of culture, the donors of the site, and indirectly, the public itself. The parties have various involvements in the process and control different aspects of the issue, yet multilateral negotiation processes, possibly including other state institutions and even political parties may take place before an issue is ascertained.

Our exemplary architect has personal considerations such as subsistence, building and maintaining a practice, and expectance of success. The motivations appear to be an important mechanism for the human designer's designing process. Although some of these motivations seem to be peripheral for the design task, they are frequently what propel the design process.

Library is a different typology for our architect, compared to her usual commissions, which mainly concern refurbishment and interior design. Thus, the situation appears to her to be a nice opportunity. She becomes eager to apply her architectural skills

that she had been developing for some period; this may turn out to be a prestige building for her and may give way to further commissions.

If our architect waits for the main issues to be ascertained before starting the design process, she would risk keeping herself out of the decision cycle. Instead, she moves forward; she has to collect, elicit, and construct information regarding the situation. More importantly, she has to construct an understanding, that is, a framing of the situation. This will enable her to propose preliminary designs to the decision makers. Through her proposals, she will start to participate in the collective construction of the decisions, which also increases her chances for the commission.

Our architect also has some preconceptions about the situation. She presumes that the rate of library usership is low in the country. For her, libraries are mostly desolate places with limited book collections, where adolescents gather for their homeworks. She thinks that this situation makes these libraries virtually unused, especially during the summer period. She also thinks that the summer period is too hot, such that the people would not intend to go to the libraries. She has not verified these assumptions by appealing to academic studies, and most probably, she will not. Nevertheless, although they might appear wrong in the end, these presuppositions will enable her to produce a first framing of the issue. She starts thinking about overcoming this supposed predicament by means of a clever design that would make the building more attractive for summer use. This may serve as a nice framing to start with, or may not; she suspends the issue temporarily.

Supporting some of our architect's suppositions, the municipality is considering a small library for mostly schoolchildren. Ministry of culture will also have a say on the final decision, because when finished, the library will be run by them, which further complicates the briefing process.

The building program has to be developed in a way that would render the contract beneficial for all sides; yet, the factors that influence the program are mostly undefined. Thus, our architect has to take initiative to make these factors more concrete. Although giving an estimate for the amount of books, users, employees, and sections is not necessarily the first problem to be tackled, our architect starts an early inquiry, in an attempt to get an overall feeling for the scale of the building.

Nevertheless, the ministry has some guidelines and quantities regarding similar libraries based on past experience, and these are conveyed to our architect.

This data may be a fine point of departure, but our architect knows well that it would not be wise to accept these preliminary numbers as hard facts, because they are often open to negotiation. The hard facts might have helped constrain the problem, yet our architect would prefer self-imposed constraints, that she had been developing as part of her practice.

2.2.3 Different viewpoints

The problem still appears as a multi-leveled and multifaceted mess. Giving some preliminary structure to this mess is still the first thing to be done. Each participant side develops its own perspective of the problem. The municipality desires a small expenditure and a building that could be presented to the public for political aims, envisaging the architect's role from this perspective. The ministry desires a building that is compliant with its standards and codes, which could be run on low energy costs and with a minimal number of staff. On the side of the architect, the basic structuring is highly subjective and depends on the worldview and professional stance.

Inspecting the relevant building codes, our architect observes that they appear highly restrictive. However, she has experiences that showed her that these restrictions could be circumvented with clever inventions and this idea appears rather exciting to her. She also discovers at this stage that she would like to take this challenge in a manner that would help her manifest a personal style. There are external factors for this decision, such as news about a schoolmate that won a prestigious competition. She wants to be successful with her design as well. Therefore, she is now intending to approach the problem as a challenge for innovation, at least one that would bring her in the spotlight of local architectural community.

Apparently, this conviction is a commitment to the unknown, which strengthens the unique character of this particular design situation, and does not help much in reducing the vagueness of the situation.

2.2.4 Design actions change the situation

Our architect starts to examine the site to better understand the important issues regarding the situation. She obtains site maps and pays several visits to the site. She takes pictures, shoots videos, and tries to detect important issues on the site and in neighboring plots. She takes notes about these and sketches a traditional building, which is nearby. The site has a mild slope with a constant descent. The main road is on the higher side and it is noisy. According to the municipality's plans, public park and library will be placed side-by-side facing the main road.

Our architect starts to think about recessing the library building from the main road. She would like to propose a new placement for the library not on the side, but within the park, which is different from the municipality's intention. However, the park has not yet been built and she starts to think that she could try to convince the decision makers that her idea would be more beneficial. This illustration reveals the unfixed character of the situation. There is even a chance that the few established decisions might be revised.

Our architect considers moving forward with both scenarios in parallel. She is aware that she could not carry this on for a long time, because this would be costly in terms of labor. This means, she has to carefully control the degree of effort that is being spent on each scenario. She will have to find the most decisive issues and focus mostly on these, leaving other issues undefined or suspended for this stage.

She calls the municipality to talk about the issue. She is informed that there are legal procedures and it is not easy to change the established status quo, yet if the mayor is persuaded, there could be a chance for revision. The meeting with the mayor could be held a week later from that time. Although this is an external constraint, it involves a critical decision for the design process.

2.2.5 Precedents and the co-evolution of problems and solutions

While the vagueness of the problem continues, our architect delves into a typological examination and searches out for library buildings and library organizations. She searches through magazines, books, and the internet. She remembers to have designed a small library as part of a larger project before starting her own practice. She remembers some of the information and solutions, some of which could be applicable in the current situation as well. She focuses on these known issues, and

possibly leaves out some other important issues, but it is not possible to gather all relevant information within this short time span.

At the mean time, she has already started sketching overall formal studies about a facade, trying out stylistic ideas that have stuck in her mind. An image of an interior scene makes her sketch about furniture and study the interior ambience. She works on combining some building parts in some certain formations following an image seen on the internet. These actions start to unroll loose threads in her mind.

While our architect is lingering on her process, she is called back from the municipality and informed that the meeting with the mayor would be held a few days before she was expecting. Because this is the most critical meeting for her to get the commission, she does not like to appear at the meeting without preparation and she feels obliged to produce a preliminary proposal. Additionally, she would like to try her chances for a replacement of the building plot and a convincing proposal might help for this aim.

2.2.6 Dynamic utilization of prefabricated strategies

Our architect starts to work for a convincing proposal that would prove both her competence and the strength of her ideas. The task is already very hard but there is also a shortage of time. Compared to the difficulty of the task, the resources are always in shortage.

In this quite vague situation, the architect has to find out where to start, what to design, what to present, and in which level of detail. These questions are what characterize the situation, and there is no reliable standard collection of answers to these questions. In each unique situation, the answers would be different.

Nevertheless, our architect answers them, perhaps just because she has to. Indeed, she already has ideas regarding these questions. Starting from her school years, she has gained experience and has developed a collection of prefabricated strategies. Although the strategy she has chosen had not been validated by her for similar cases, in a rather strategic move, she accepts it as her first certainty. She uses this strategy only to start the process. Further on, she will adapt her strategy according to the needs of the particular situation. Indeed, at each juncture, the answers and questions will change, and she will feel the right moments to transform her strategy step-by-step through the process.

Our architect decides to focus on only the two main issues in her presentation. With this decision to limit the presentation to essentials, she produces two section sketches for both of the scenarios, which illustrate the relationship of the main road, park, and the proposed library. The building is represented in an abstract manner. After several trials, she draws presentation versions and not being content with its impressiveness, places the sketches over photographs taken from appropriate angles. Now the process starts to take the character of a graphic design task. She has to appreciate the overall visual quality of her presentation images, which would testify for her designerly qualifications, as this might be important in the establishment of the trust relationship between her and her client.

In addition to the above-mentioned functional points, our architect decides to present an overall vision for the project, which would give a vague sense for the final appearance and style of the building. She first considers preparing a 3D rendering from the entrance side, but when she starts to model the facade, she finds out that the unknown issues make it hard to model the masses and the result is not persuasive. After her first trial, she goes back to her sketchbook and prepares a sketch drawing for a hypothetical facade, where she focuses on the overall appearance of textures, materials, and colors.

Our architect knows that these decisions are at best provisional; however, the important point at this stage is to prove credibility. This illustrates the contextual nature of the choice of strategies, which strongly depends on the particular situation. The architect could have chosen to present a fast-sketched whole building with sketch plans and preliminary massing. However, this strategy would risk being perceived as an incompetent designer. Although it would be normal to present half-baked ideas at this stage of a design process, this would risk losing the commission. Instead, our architect focuses on the details that would normally stay undecided until the end of the construction process.

2.2.7 Contextuality

The above illustration coincides with Schön's (1983) claim regarding the characteristics of design situations, i.e., uncertainty, instability, uniqueness, and value conflict. As was discussed in previous sections, the fact that the real world problems exhibit these characteristics prevents establishing direct links between the

theoretical area of design research and practical applications. This is mostly because the theoretical knowledge in such a field has to stay essentially incomplete. Designers operate in a situated manner, constantly reformulating their problems with regard to the feedback gained from their context. On its own, design knowledge is vague and open-ended; it finds determination only within specific contexts.

Assume that there is an autonomous intelligent agent, which aims at dealing with early phases of architectural design. The main problems for this agent would be to understand the situation thoroughly, to bring in some motivations to move forward, and choose the right strategies for producing and communicating. This clearly requires a general form of daily intelligence in addition to a domain-specific understanding of the context.

Our architect is not equipped with an objective type of knowledge but she chooses, builds, and endures a possibly consistent viewpoint. This is a key pattern in designer behavior (Lawson, 2005), because, the designer needs a personal perspective to be able to start, propose, and operate. She needs to find at least one working strategy; not all the possible methods for designing, or the best one, but at least one. Her methods are unique and unsystematic; they are prepared just on time.

2.2.8 Constraints, objectives, and criteria

Our architect attends the meeting with the mayor and presents her arguments and proposals. The mayor appears sympathetic for her proposal for site replacement and he promises to deal with the issue; however, he also points at the hardships of such revisions. Although not impossible, the chances are not so high, he asserts. The new situation is rather unpleasant for the architect, because as the ambiguity on the decision lingers, it will be more expensive to continue on two alternatives simultaneously. If she chooses to continue on only one of these alternatives, it would also be expensive to switch from one alternative to the other.

At the meeting, other issues appear about the facade images. The mayor appears lukewarm towards the proposal's style. He bases his arguments not on his personal taste but on some social values regarding the traditions. He asserts that he could have been more sympathetic with a design that emphasizes traditions and national culture, though he is in favor of modern buildings. The architect inspects her own situation against this approach and briefly considers whether she could seek a compromise

between her favored style and traditional techniques; however, she comprehends that she is fervently devoted to her modernist attitude and leaves this issue suspended. She will have to reconcile these normative and aesthetic issues somehow, either by convincing the mayor or by taming her own stance.

Nevertheless, the mayor is mostly positive, not because of the proposal, but because of the sympathetic character of the architect. The contract is assured and our architect can continue working on the problem. She has to develop the technical drawings, preliminary details, and cost estimates.

2.2.9 Dynamic structuring strategies

The complexity of the product of architecture appears at this stage. A building is composed of a multitude of subsystems, including layout, circulation, structure, energy, HVAC, and envelope systems. Each of these systems depends on the others and it is not easy to decide on one until the end of the process. All these systems are also dependent upon other factors including cost and style.

Our architect studies each system both separately and together with the others. She knows well that the decisions at early junctures are not final and are subject to reconsideration, yet she makes provisional decisions and moves forward to reconcile requirements of each of the subsystems with respect to the overall constraints and objectives.

According to her hierarchical planning, our architect first studies location and massing. At this point, she realizes that she could try to design a proposal that satisfies both building location scenarios simultaneously. This is another challenge, but she is eager to try it. While she is studying overall issues, she also carries forward several ideas regarding details. They will have to be incorporated into the same building one by one. She continues studying the problem on several different scales and abstraction levels. She uses diagrams, sketches, and texts for her abstractions, and drawings for physical studies. For each of her studies she has to consider several issues, but it is up to her to choose which issues are more urgent and important at that point through the process.

More often than not, she works in an integral manner. She studies a section of the building where she simultaneously considers the relationship of the main road, entrance, vertical functional relations, front facade, day lighting, and ventilation. She

also occasionally works on the facade, but these studies are interrelated and each move possibly necessitates a change on another aspect, even though changes are not propagated immediately. She has to think about the disposal system while placing the lavatories, which might be considered together with the vertical elements, which is connected to the entrance. Whenever they appear, she tries to reconcile conflicting issues in pairs and does not hesitate to suspend a decision when she is stuck. She sometimes fixates on some decisions, but ultimately she knows that there is no decision that could not be revised.

All through this process, our architect has to carry forward a strategical and tactical reasoning as well. She mostly does this in an intuitive manner. She has to decide on which scale, with which techniques, and on which issues she would continue her study. She has to decompose both the problem and the process in a flexible and dynamic manner. She also needs to re-integrate these decomposed parts into coherent wholes and has to ensure a unified flux of parallel studies. After each move, she evaluates aspects of the new state. She also has to choose which criteria, constraint, or objective is important at that juncture.

2.2.10 A multitude of possibly conflicting objectives

There is an essential complexity in a design situation in terms of multiple objectives. Most of these objectives are not independent from the others and are often in conflict with each other. Consider the dimensions of a south facing window, which can provide sunlight, natural illumination, and ventilation with the cost of worse heat and sound insulation. There might be an optimum degree for each of these objectives, which has to be determined for each particular project. When constraints are considered, although they give structure to a problem, sometimes they make it harder to obtain desirable results.

While the project unfolds, regular meetings are held and the proposals are negotiated between parties. There is a tight deadline for the design phase and several details have not been appropriately studied according to the architect. The other decision makers however, demand the project to be submitted so that they can start the bidding process for the construction phase. Our architect is ensured that it is usual for final touches to be done during construction and that she can refine finishings and details during this phase. Although this can change some issues considerably, the

architect has to agree, because the contractor would want to decide on subcontractors, which will affect the procurement and detailing of subsystems.

There has to be better estimates of cost at this stage. There is a preliminary proposal, which allows the specification of the quantities and details of basic productions and materials. The task seems to be rather well-defined and the task environment is such that even an autonomous artificial agent could search for information regarding alternative systems and materials over the internet. Although there is always a margin of negotiation on the prices, such a task seems to be a probable candidate for automation. However, the problem is by no means trivial. Consider the problem of keeping up with a cost threshold. Whenever the total cost exceeds this threshold, the agent has to find cheaper alternatives for some of the productions. However, the problem is not as simple as deciding amongst equal alternatives. Each provider's services and products will have advantages and drawbacks, and a comparison between two similar products may become complicated. Moreover, in many cases the design has to be revised, instead of simply choosing cheaper materials.

In our story, the total cost that is proposed by our architect is found to be too high and the negotiation focuses on a large entrance canopy that the architect finds most important, socially and functionally as well as for air-conditioning. The architect has several valid arguments but she still has to find ways to cut down the overall costs. She might revisit every section of the design to diminish the cost of each part or just find a cheaper way to construct the canopy, which would not spoil its appearance. As a solution, she separates the canopy into pieces, so that it only covers the necessary areas. The appearance is considerably changed; it is not as pleasing to her as before, yet it still fulfills its functionality. In this case, the costs necessitated a reconsideration or a partial reframing of the situation.

It is not straightforward to automate the task that concerns a balancing of the costs. This is because, the task is also linked to several other aspects of design. The cost agent has to be able to appreciate the construction market and foresee further design possibilities. Another problem here is, it is not easy to ascertain the materials and subsystems until construction phase begins. However, if the decisions have already been taken and fixed, then it is straightforward to carry out the quantity surveys, as is done with building information modeling (BIM) applications.

This example illustrates the claim that, in architectural design some well-defined and ill-defined procedures are often intertwined. There are better-known objectives like keeping up with structural and dimensional criteria and constraints brought forward by building codes, standards, or aspirations such as LEED[4] or BREEAM[5]. These can be targeted by semi-automated sub-procedures to be operative together with the intelligent agents, i.e., within human-tool complexes for more efficient and effective design processes. The intelligent agent is still necessary here to carry out the procedures that require general intelligence, such as normative and aesthetic issues, communication and negotiation, dynamic decision making and decomposition – re-integration etc.; in short, for dealing with the holistic, undefined, and uncertain aspects of the process. This ends our illustration and brings us to the issue of artificial design intelligence.

2.3 Computational Design

Initially understood simply as the procedure of calculating (i.e., determining something by mathematical or logical methods), with the advent of electronic computing machines, computing has come to be understood predominantly with regard to computing devices, i.e., as any goal-oriented activity requiring, benefiting from, or creating computers. Conformably, the term computation is the act or process of computing, where complicated calculations are carried out through an algorithm or a protocol.

With a broader definition following an information-processing approach, computation can be defined as producing the appropriate output, given input and processing facilities. This definition attributes the intermediary stage, i.e., the processing stage to computation. An even broader variant of this definition can be stated as, computing is any causal progression, given a consistent computing medium, which processes input to produce useful output. Note that the terms "appropriate" and "useful" within above definitions point towards the goal-oriented

---

[4] Leadership in Energy and Environmental Design, "a voluntary, consensus-based, market-driven program that provides third-party verification of green buildings" (Url-1).

[5] Building Research Establishment Environmental Assessment Method, an "environmental assessment method and rating system for buildings" (Url-2).

aspect of this process, which, by ruling out random physical phenomena that are not related to the perception of a goal directed agent, narrow down the definition. Similarly, the terms "processing facilities" and "consistent" refer to the computing media, which exhibit some sort of temporal continuity, hence implicitly referring to an at least partially autonomous computing system.

It should be noted that, these last definitions are not solely mathematical or limited to computer science—the disciplines that computation is usually attributed to. Indeed, many physical and biological processes can be characterized in algorithmic terms. This may be seen as giving license to speaking of all physical or biological systems as computers to such an extent that with a pancomputationalist approach, which carries this broadness of definition to its extremes, it might be claimed that anything that is subject to causal progression is computation. This is why a cautious limitation is attempted when giving definitions of computation.

Ironically, the broadening of the meaning of computation is mostly due to the advent of computer science. Faced with many different types of computing environments, like sequential or parallel digital computers, cellular automata, quantum computers, neural networks, and analog computation the meaning of computation inevitably expanded within the philosophical realm and the computer metaphor replaced the previous mechanical one for intelligence.

As another consequence of this broadening of the meaning of computation, "the computational theory of mind" has been brought forward by Hillary Putnam for understanding and explaining brain in terms of computation (Horst, 2011). In a similar vein, information-processing theory has set out to establish a parallel between the symbolic computing environments and the human brain (Newell and Simon, 1972). Computational theory of mind has two essential components. The first is the representational theory of mind, where mental states are held to be representational in the sense of including, as constituents, symbolic representations having both semantic and syntactic properties (Horst, 2011). The second component is the computational account of reasoning, according to which, reasoning is taken as a process in which the causal determinants are the syntactic properties of the aforementioned symbolic representations (Horst, 2011).

It is obvious that human minds are able to take information as input and process this information. This reveals the human mind as an information-processing system. Yet, it does not seem necessary to liken a human mind to a digital or other type of computing machine for such conception. It is not an aim for this thesis to delve into related questions. These somewhat broad conceptions of computation are mentioned here with the purpose of indicating a possibility to comprehend operations of a human mind and a computing machine as residing within an operational continuum, which enables a human-tool couple to be conceptualized as a kind of unified computing medium. An effective coupling of human minds and computing machines is exactly what Computer Aided Design (CAD) is about. Therefore, it is a basic interest of Computational Design studies, which aim at developing CAD tools and methods.

From its inception, studies of the field of Computational Design have been closely related with the advances in AI. The basic assumption was that, intelligent computational systems could aid human designers better. Experimental computational techniques borrowed from AI studies have been put to test for almost every aspect of design and this went hand-in-hand with the studies that aim for a more scientific or systematic study and practice of design; as is shown by the brief history of the first generation of design research (Cross, 1977; Bayazit, 2004; Cross, 2006).

It is not easy to ascertain the border where AI starts within Computational Design; however, it can be claimed that the level of intelligence within an artificial system can be assessed by the degree and success of involvement and autonomy of the non-human systems when carrying out specific tasks. Kalay (2004, p. 419) suggests three discrete levels for classifying CAD systems in terms of their intelligence:

1. Tools, which must be instructed line by line, like computer-aided drawing systems.
2. Assistants (semi-autonomous tools), who can take instructions and carry out several steps, e.g., an expert system.
3. Intelligent design agents who are completely autonomous while carrying out their intended tasks.

Technologies on the first level are commonplace (e.g. drawing and modeling applications), and there are working examples of the second (e.g. some functionalities of Building Information Modeling systems, i.e., BIM[6]). However, the third level is currently reserved for human designers. An attempt to attain the third level has to cope with a complicated task environment, which requires extensive mental skills together with a human understanding of design situations. Therefore, CAD takes place in a design environment where humans and CAD tools operate together, constituting varied forms of the aforementioned human-tool complexes. The role of the CAD tools is usually taken as aids to the human designers, who act as the principle managers of the process.

One frequently stated aim of such cooperation has been to relieve the human designers from the burden of repetitive and tiresome tasks on which the computers excel; therefore, to increase efficiency and pleasure by allowing humans to focus more on the creative aspects of design. However, this conception of the computer as an insensitive calculator has been challenged very early on by one of the pioneers of Computational Design. Nicholas Negroponte, whom at once shared this traditional task division between human and computers, later converted to an alternative position (Negroponte, 1975, pp.8-13). During 1960's and 1970's, Negroponte's Architecture Machine Group at Massachusetts Institute of Technology explored methods for a close companionship of artificial agents and humans. According to Negroponte (1970; 1975; 2011), in order for them to aid human designers properly, artificial agents had to have a sensitive intelligence that would enable them to understand and respond to the humans.

[6] BIM is a commercially successful and promising area of computational design, which combines parametric modeling technologies with a set of processes to produce, communicate, and analyze building models (Eastman et al. 2011, p. 16). BIM building models are characterized by: (1) Building components that are defined with digital representations (objects), which carry computable graphic and data attributes that identify them to software applications, as well as parametric rules that allow them to be manipulated in an intelligent fashion. (2) Components that include data that describe how these components behave, as needed for analyses and work processes. (3) Consistent and nonredundant data such that changes to component data are represented in all views of the component and the assemblies of which it is a part. (4) Coordinated data, such that all views of a model are represented in a coordinated way (Eastman et al. 2011). Thus, BIM platforms offer the ability to incorporate a type of specialized knowledge related to the components of a design environment.

The demanded sensitivity on behalf of the artificial agents has been an ongoing challenge for Computational Design studies. With the advent of personal computers, the co-evolution of the tools and human agents has entered a faster phase. Tools such as ruler and compass that had been rather stable in their use and physical constitution for over hundreds of years, have now been mostly discarded, while constant change is intrinsic to their substitutes, that is, the new computational tools, which are regularly being upgraded. As a result, the roles of the agents and their preferred techniques are now under constant transformation.

Today, CAD tools extend over a broad field comprising design and representation tools for drawing, modeling, and animation, as well as evaluation tools for analyzing acoustics, structure, illumination, energy efficiency, cost, materials, evacuation, etc. CAD tools are being used for design, optimization, information storage, communication, collaboration, construction, and management.

It is possible to classify CAD tools over a series of scales. For example, as stated above, on a scale of intelligence, tools like drawing applications appear to lean towards the mechanical end, while more complicated platforms like BIM systems may be placed relatively closer to the intelligent end. There may also be a scale, which locates tools according to their levels of generality. A technical simulation tool would reside on the specific (or expert) end, while a modeling and animation platform may target a general audience from all arts and design fields. Likewise, on a scale of flexibility, solid modeling tools oriented for engineering tasks may be contrasted with scripting environments through which users gain the power to extend the capabilities of an underlying platform. Most drawing and modeling platforms provide some kind of scripting facility for this purpose.

Through scripting interfaces and parametric modeling techniques, generative and parametric approaches have found their way into the innovative designers' repository. While most computation is hidden from the end-users in popular CAD applications, in generative and parametric approaches further potentials of computation are being exploited, often through custom-made algorithms.

Generative techniques start with a limited set of rules or procedures that define syntactic articulation of given items and aim at producing a variety of proposals for a problem. Parametric approaches, on the other hand, mostly appear as integrated form

generation and evaluation methods and although not necessary, in many cases they take the form of one-off idiosyncratic design strategies. Development of such a method frequently is a matter of an iterative process, which coincides with the method's application. Therefore, a parametric design process unfolds through a tight integration of problem, solution, and strategy. In a sense, each such definition involves a kind of embedded intelligence, which becomes indistinguishable from the design situation that gets to be well-defined through the use of the method itself[7].

Not only generative or parametric approaches, but all CAD tools, by definition, involve platforms supported by computational means. Indeed, it is not mere computation, but also a kind of intelligence, which is present in many CAD tools. For example, modern rendering engines like V-Ray (Url-3) or LuxRender (Url-4) embed stylistic decisions. With simple parameter settings, these rendering engines are able to submit finely illuminated and shaded 3D images that considerably ease a graphic artist's job. Likewise, a digital drawing tool like AutoCAD (Url-5) could be seen as a computational assistant. Firstly, it guides the user with its interactive rulers, compasses, and other tools, and secondly, it carries out a great many tedious calculations, while the user concentrates on her drawings. It helps the user to catch line corners and intersections, and copy and store entire or partial drawings to make them reusable. The user might be giving line-by-line commands to the program; nevertheless, the program executes a series of complicated steps after each command. In this sense, even a drawing application, which is not mostly thought to include intelligence, operates through a precious type of intelligence; and is successfully incorporated within a human-tool complex that greatly increases efficiency. This is also valid for modifier commands of 3D modeling applications. These modifiers are shortcuts for complex operations that are mostly parametrically controllable to some extent.

At first sight, operations of these tools may seem trivial. However, if a human tries to carry out these tasks manually, she would understand the scale and difficulty of the problems. It should be noted that many mathematical procedures that seem trivial just because they are subject to algorithmic solutions are products of the cumulative

[7] For developments and discussions related to computational techniques in architecture, see Kolarevic, 2003; Spiller, 2008; Littlefield, 2008.

effort of the brightest minds of human history. They are the most refined results of an important constituent of human intelligence, namely logic, mathematics, and systematic thinking. Moreover, it was this line of research, which culminated in the very idea of computing and ultimately in computers. It is not, then, surprising that the first generation AI studies had been developed within a paradigm according to which these aspects of human thinking were being held to be the highest and most difficult of all. Nevertheless, this conviction has been disproved by the futility of the same line of research for AI. It was exactly this type of systematic and generalizable thinking, which was proved ineffective on many occasions for ill-defined real world tasks such as encountered in architecture.

In complicated design practices like architecture, it is hard to find tasks that could be totally systematized. However, it is equally difficult to find tasks that are totally devoid of computational processes. Indeed, humans work together with computational systems on all tasks distributed to all levels of a design situation. Today architectural design is a mixed initiative, where computational procedures and human design expertise intermingle to generate a unique character for one-off design processes.

This is not a new occurrence though. For example, rule-based production pipelines have been witnessed and recorded in architecture since antiquity. Here the sequence of rules indicates a kind of systematic and general way of carrying out tasks; and remind a more or less mechanical procedure. Rule-based approaches can be found in the Renaissance period as well as in traditional architectures. According to Vitruvian tradition, a kind of rule-based—and in some cases parametric—architecture was responsible for basic architectural typologies, in particular for the relative proportions, placement, and quantities of some building components, or features of these.

The procedural knowledge in traditional architectures is often transmitted through selective rule sequences, to be applied in interaction with a new context. Starting from plan alternatives, developing these alternatives according to a site's conditions and user requirements, placement of the basement, the load bearing elements, partitions, flooring, how each part would be connected with all the others, hence structural details, roofing, and ornamentation all appear to be coded with a limited set of rules, which are somehow flexible to enable adaptation. Some of these are

expressed through explicit rules, heuristics, or norms, while some of them are stored and transmitted as practices, and are gained in a tacit manner, through a master-pupil interaction.

The rule-based, heuristic, or propositional aspects of human thinking and knowledge are transferred relatively easily into generalized tools that help apply these rules, as in the example of ruler and compass. The former constrains the pen to a linear direction while providing discrete dimensional steps inscribed on it, and the latter has a continuous range in the radius dimension, yet constrained by physical conditions to draw only circles.

On the other hand, tacit aspects of design knowledge tend to be linked to the very functioning of the human nervous system whose information processing procedures have not yet been completely mapped. The requirement of tacit knowledge is also strongly related with how a task is being carried out. Invention of alternative procedures to deal with a task sometimes helps dispense with a need to understand how humans carry out the same task. For example, usage of a hammer is enabled by a relative stability and uniformity of the physical requirements for each specific nail to be thrusted into each specific material. However, the usage of hammer and nails is also strictly dependent on human anatomy, and learning how to use them has a tacit dimension. On the other hand, within workshop conditions, or on a production line, it is easy to envisage a mechanical hammer whose steps could be explicitly described and listed. This change in how the actions are carried out dispenses with the need for a tacit dimension.

The tool to help engrave the precise shape of the fluting around a Doric column might be the combination of compasses, rulers, ropes, chisels, plumb lines, a series of written rules, a series of known heuristics, and the masters that operate these tools and manage the process. This combination of tools and humans in turn can be considered a higher level tool itself; or better, a human-tool complex, thus indicating the historical continuity of today's mixed initiative design practice, where digital surveying and computing equipment mostly replaced the analog ones like the ruler and the compass.

It can be observed in the above examples that, some of the knowledge has been externalized to the tools like compass, ruler, and plumb line. Within a human-tool

complex, some of the rules and constraints of possible actions are embedded directly within the physical production environment. What this amounts to is a degree of externalization of the knowledge and skills, a partnership with the tool and the human, which on the first hand, may reduce the need for human involvement, and on the second, may enable new productions that have not been possible solely by human capabilities. Explicitation of a series of rules are only one side of the externalization of the productive knowledge, the more important side is the new capabilities gained by this ever-renewed human-tool collaboration. Each manner of cooperation between humans and tools has the possibility to extend the capabilities of the human designer, with specific advantages and drawbacks. Drawing on the sand helps calculating and seeing possibilities, while direct application on the site requires knowledge of construction procedures and skills to adapt these procedures to new contexts. In comparison, drawing on paper eases the burden on the human memory and enables more accurate, detailed, and durable information storage. The means to carry out real world tasks are developed according to the answers to the questions concerning the kinds of tasks, contexts, agents, tools, and procedures.

As is the case with the above examples, in any task environment, a kind of continuity appears between tools and agents. Each tool may obtain a kind of agency and may tend to impose its will on processes. In these cases, it becomes hard to draw strict borders between an agent and a tool; especially between the non-human tools that constrain processes and the tools that are regarded as the very parts of a human; specifically, the mechanisms that are related to the mind, like perception, speech, memory, vision, or reasoning. The difficulty is, we can discern the tools or the humans as agents, but we can denote constituent human faculties as tools or agents, too. In brief, with respect to computational systems and human agents, CAD refers to "an ecology of mutual design complementation, augmentation, and substitution" (Negroponte, 2011) and understandings concerning the roles of computational tools and human agents are also due to interpretation and change.

2.3.1 The extended mind and the human-tool complex

In his book, Consciousness Explained (1991), philosopher Daniel Dennett reminds us how our perceptual sphere naturally extends through external objects. When we grab a pencil and rub its tip on a series of surfaces, we feel the textures of these

surfaces as if we have sensors at the tip of the pencil. The same is observed when within a car we pass over a patch of oil. We feel the greasy surface as if we have sensors under the tires of the car.

This natural extension is not limited to perception though. In their article on the extended mind, Clark and Chalmers (1998) show how readily the mind is extended over external tools to increase its effectivity. They give examples like the use of pen and paper to perform multiplications or the use of instruments such as the slide rule. In these cases, while the human brain performs a series of operations, another set of operations are delegated to the manipulations of external media. The point here is not just the presence of external computing resources, but rather the general tendency of human reasoners to lean on environmental supports (Clark and Chalmers, 1998).

For example, in the Tetris game, where falling geometric shapes must be rapidly directed into an appropriate slot in an emerging structure, a rotation button for physical rotation is used not just to position a shape ready to fit a slot, but often to help determine whether the shape and the slot are compatible. This is an action that alters the world to aid and augment cognitive processes such as recognition and search. According to Clark and Chalmers (1998), rather than being passive objects of contemplation, the relevant external features in this process are active elements and they play a crucial role in the process. Because they are coupled with the human organism, they have a direct impact on the organism and on its behavior. Clark and Chalmers (1998) call this position 'active externalism':

> In these cases, the human organism is linked with an external entity in a two-way interaction, creating a coupled system that can be seen as a cognitive system in its own right. All the components in the system play an active causal role, and they jointly govern behavior in the same sort of way that cognition usually does. If we remove the external component the system's behavioral competence will drop, just as it would if we removed part of its brain. Our thesis is that this sort of coupled process counts equally well as a cognitive process, whether or not it is wholly in the head (Clark and Chalmers,1998).

This extension of the mind is also related to the situated character of cognition. The situated agent is in continuous transformation in response to the effects of an external environment. This process seems to occur via mechanisms that are similar to the ones at play in the aforementioned extension of the mental processes. Although they are seen as independent systems, humans perform not only within, but also together with their environments in a tightly coupled manner. The perceived immediacy of

this coupling should be considered as a factor in the development of computational tools.

In design, an important use of tools is their functioning as external memory. Human memory is particularly good at classifying, learning, retrieving, and matching patterns, concepts, and episodic sequences; however, usually not as successful in remembering a sequence of numbers or all the minute details of a physical arrangement. A drawing is a type of external memory that can be used to remedy this defect of the human mind. However, there are other important uses of a drawing. First, it helps the designer to produce a more concrete reality according to some vague initial ideas. Secondly, it helps her to see what she is producing. Thirdly, in its technical forms, it helps her to communicate these emergent ideas in an unambiguous manner, which is very important at least for modern production processes.

Some of the design tools help designers calculate, measure, and evaluate. Basic examples are the compass and the ruler, which help the designers to locate points and geometric objects inside two or three-dimensional spaces without explicitly delving into complex calculations. In a sense, the tools carry out required calculations. Thus, the human-tool complex inside a real world situation is akin to a computing environment. How this complex of human and tool operates is among the concerns of this thesis. Indeed, as has been seen, the degree and style of the involvement of each participant within this complex is instrumental in defining CAD processes. The critical point here is to head towards the tools that would effortlessly become a natural extension of the human agent, allowing for effective, productive, and pleasant production environments.

Today, emerging 3D visualization, augmented reality, and body gesture based controller technologies provide a possibility for more natural and pleasurable human-computer interactions. On the other hand, non-intrusive body-mind additions, e.g. spectacles, have been in use for quite a long time, which supports the idea that humans have a natural capability to be extended and an interface does not necessarily demand advanced technology. A simple sketch, with a piece of tracing paper laid over it may as well be seen as an interface, which establishes a bridge between external memory and a human agent.

2.3.2 Computational Design as a research field

Since CAD tools are now commonplace, and virtually no design process is left without computer aids, it seems plausible to gather virtually all design practices under the title of CAD; in which case design and CAD become interchangeable terms. On the other hand, the term Computational Design, before denoting a type of design practice, can be taken as indicating a research field. Thus, it can be claimed that CAD is simply the current design practice, which makes use of computational tools, while Computational Design is a field of research, which searches for better and more advanced CAD, through new methods, techniques, and tools.

It is the development of computational tools, rather than designing with these tools, which distinguishes the field of Computational Design. Consequently, after being absorbed within daily practices, basic usage of a computational technique or tool is no more seen as part of research; the frontier of the field is constantly being shifted. Thus, a broad definition of Computational Design can be given as the development of computational approaches, methods, techniques, and tools, to better understand and enhance design.

With its methods and tools, Computational Design has an important effect on the transformation of design processes. By developing new techniques and tools that enhance the ability to study, represent, analyze, and produce complex formations, it changes designed artifacts. In addition, through all these transformations and by providing new metaphors, it influences design culture in general.

We can discern two main areas under Computational Design. The first appears within practice-led-research (or research-by-design), mostly in the form of one-off, problem-specific methods and tools. The second area aims at developing reusable approaches, methods, techniques, and tools, and although these might be developed and validated through practical applications or through research-by-design methodologies, the primary orientation is towards research rather than direct practice.

As forms of practice-led-research, generative and parametric approaches often overlap with the area of Computational Design. These techniques carry a potential for bringing new possibilities in form generation as well as a more developed ability for the production of intricate forms. Moreover, they bring the roles of human agents

and tools under continuous questioning. Hence, they exhibit a potential to change the understandings and processes of design and they are being used with the effect of expanding creative fringes within the borders of design fields. Nevertheless, it remains a more basic task for research fields to develop generic, flexible, and reusable approaches, methods, and tools.

It can be claimed that a certain threshold has been reached in several areas of CAD tools. Drawing, modeling, and graphic design applications are now reliable and productive as representational tools. On the other hand, intelligent design support for creative aspects of design has been relatively neglected. Therefore, a potential expansion area for CAD appears as artificial design intelligence, i.e., embedding intelligence within CAD tools for better and more productive human-tool interactions, not only for detailing and optimization phases, but for creative tasks as well.

It is now widely assumed that many tasks within design processes are not solvable by simple algorithmic or heuristic programming, or through logic or rule-based systems. However, it is apparent that human agents are able to handle such tasks and if we are not to attribute mystical powers to human beings, then we have enough reason to continue this line of research. These considerations bring us to the question of artificial intelligence in design.

2.3.3 Artificial intelligence in design: information-processing, problem-solving, search, and formal approaches

Attempts at building intelligent entities have started soon after World War II, and the name artificial intelligence (AI) has been coined in 1956. Russel and Norvig (2010, p. 2) cite several textbook definitions for the field of AI. In combination, these definitions refer to the development of systems that act or think, either rationally or as humans do. Note that definition of terms like 'rationality' and 'thinking' are usually left to specific contexts and applications. Two basic motivations can be discerned in the definitions of AI. The first is the bottom-up development of particular technologies for specific practical aims, which is a predominantly practical orientation. The second motivation is rather a top down quest for understanding, mimicking, and rebuilding the human mind and its capabilities in general, or in the whole.

The first kind of AI encompasses a variety of subfields including general-purpose areas, such as machine learning, computer vision, natural language processing, and automated reasoning, as well as specific tasks like playing chess and backgammon, proving mathematical theorems, writing poetry, and diagnosing diseases (Russel and Norvig, 2010, pp. 1-2).

The second kind of motivation has brought forward attempts to understand and mimic human mind by way of comprehensive neural mappings, brain simulations, cognitive architectures, and agent frameworks often with an aim to develop practical artificial reasoning procedures and hardware for intelligent agents, e.g., SOAR (Url-6), ACT-R (Url-7), Brain-i-Nets (Url-8), Blue Brain (Url-9), Brains in Silicon (Url-10), Spaun (Eliasmith, et al, 2012), The Human Connectome Project (Url-11). At the mean time, another set of comprehensive projects such as Cyc (Url-12), DBpedia (Url-13), and WordNet (Url-14) have set out to generate structured databases of all commonsense knowledge.

The first studies in a quest for artificial design intelligence have followed the information-processing paradigm pioneered by Simon and Newell (Cross, 1977). Simon and Newell's studies on problem-solving had led them to understand it as a kind of heuristic search process (Simon, 1973). The topic of state-space search originated in the early years of AI, with Newell and Simon's work on the Logic Theorist (1957) and GPS (1961). These studies led to the establishment of problem-solving through search algorithms as the primary AI task (Russel and Norvig, 2010, p. 109). After devising computational methods to experiment with their ideas, assuming an approach, which is called cognitive simulation, they went on to claim that their studies helped explain how the mind ultimately worked (Newell and Simon, 1972).

Simon and Newell's studies and ideas have also found attention in Computational Design circles and have been taken as the practical clue for achieving design automation. Hence, design was defined as an instance of problem-solving and started to be treated through heuristically guided search methods. According to this approach, to solve a given problem, an initial task environment has to be defined (Rowe, 1987, pp. 51-52):

- First, there is a problem space whose elements are knowledge states, some of which represent solutions to a problem.
- Second, there are one or more generative processes or operations that allow one to take knowledge states as input, or as starting positions, and produce new knowledge states as output.
- Third, there are one or more test procedures that allow the problem solver to compare those knowledge states that are presumed to incorporate solution properties with a specification of the solution state that is not the knowledge state itself.
- Finally, there are further processes (i.e., heuristics) enabling a problem solver to decide which generative processes and which test procedures to employ, based on the information contained in available knowledge states.

This way, it becomes possible to consider a solution state with reference to a set of world states in which the goals will be satisfied. The agent's task is to find out which sequence of actions will get it to a solution state. Before it can do this, it needs to decide on what sorts of actions and states to consider. This problem-setting, or problem formulation action is crucial and corresponds to a distinct analysis phase, and is mostly carried out by the humans and fixed before the search begins.

In general, an agent with several immediate options of unknown value can decide what to do by choosing the best sequence amongst the different possible sequences of actions that lead to states of known value. “The process of looking for a sequence of actions that reaches the goal is called search” (Russel and Norvig, 2010, p. 66). A search algorithm takes a problem as input and returns a solution or an action sequence that leads to a solution. Once a solution is found, the actions it recommends can be carried out, which is called the execution phase. Thus, there is a "formulate, search, execute" design for the agent, where search and execution phases may coincide. After formulating a goal and a problem to solve, the agent calls a search procedure to solve it. It then uses the solution to guide its actions.

For the task of search, the states (or nodes) of a search tree have to be generated (Figure 2.8). These nodes are evaluated with regard to an objective function or in terms of its compliance to a set of constraints. Then, through guiding heuristics, available branches and nodes of a search tree are traversed to find an optimum

solution; or in some cases, an available course of action. The procedure of generation can take the form of the wholesale creation of a possible state or the modification of an existing state by several operators; hence the related expressions, ‘operator – state’, ‘generate – test’, or ‘propose – critique – modify’ (Figure 2.9).

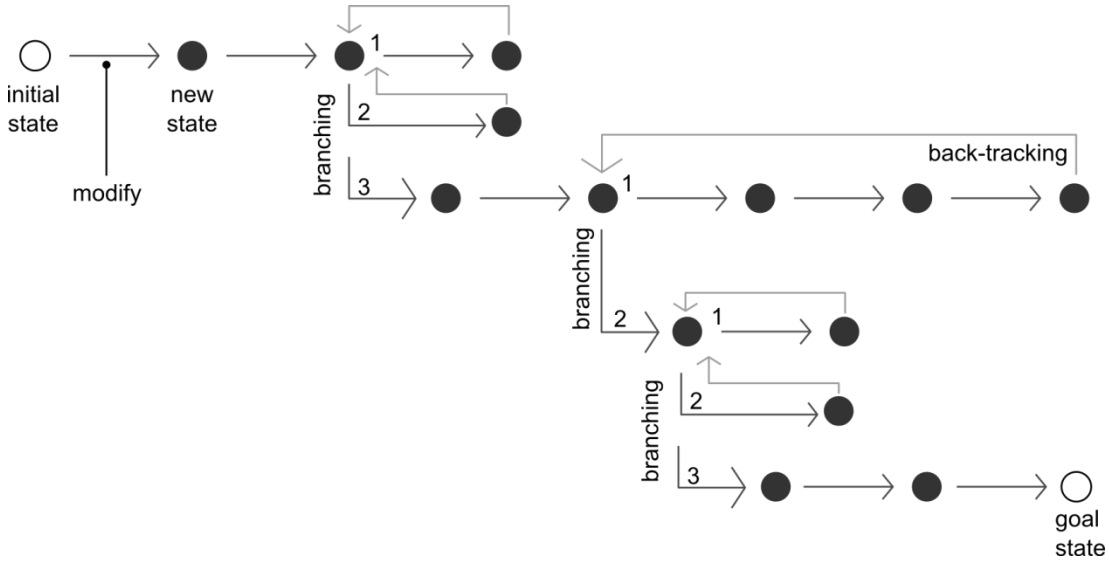


Figure 2.8 : Basic search concepts.

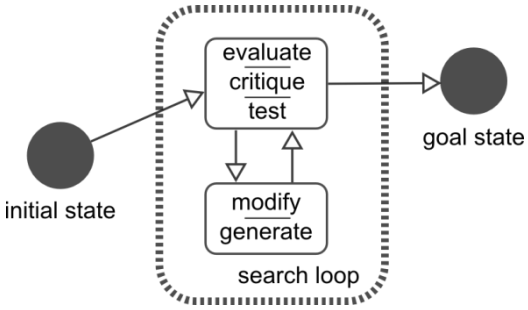


Figure 2.9 : General scheme of generate and test algorithms.

Uninformed or blind search means that the strategies have no additional information about states beyond that provided in the problem definition (Russel and Norvig, 2010, p. 81). These algorithms only generate successor states and distinguish a goal state from a non-goal state. In these cases one of the several algorithms, such as 'depth-first search', 'breadth-first search', 'uniform-cost search', 'depth-limited search', 'iterative-deepening search', etc., are used to search through the whole of the search spaces.

However, complex search problems—where the search tree branches exponentially—cannot be solved by uninformed methods, for any but the smallest instances. Strategies that take into account whether an intermediary state is more

promising than another are called informed search or heuristic search strategies (Russel and Norvig, 2010, p. 81). In the informed cases, problem-specific knowledge beyond the definition of the problem itself is utilized to find solutions more efficiently than an uninformed strategy. One or more heuristics are applied to guide the search process to branches that are more promising, or to narrow-down (or prune) the search tree (Russel and Norvig, 2010, p. 92). The general form of this type of algorithms is 'best-first search'. However, most of the times, the best node is not known in advance. In practice, the node that appears to be best according to the evaluation function is chosen (Russel and Norvig, 2010, pp. 92-95).

The above-described search algorithms are designed to explore search spaces systematically; as a result, these approaches only address a single category of problems: observable, deterministic, known environments where the solution is a sequence of actions. However, there are alternative formulations for different task settings. If the path to the goal does not matter and only attaining the final state is desired, local search algorithms can be considered instead of systematic ones (Russel and Norvig, 2010, p. 120). Local search operates using a limited number of nodes and generally moves only to neighbors of those nodes. Although it is not systematic, local search has two key advantages: (1) it uses very little memory, and (2) it can often find reasonable solutions in large or infinite (continuous) state spaces for which systematic algorithms are unsuitable (Russel and Norvig, 2010, p. 121). 'Hill-climbing' (or 'greedy local search', Figure 2.10), 'simulated annealing', and 'evolutionary algorithms' (Figure 2.11) are treated under this title.

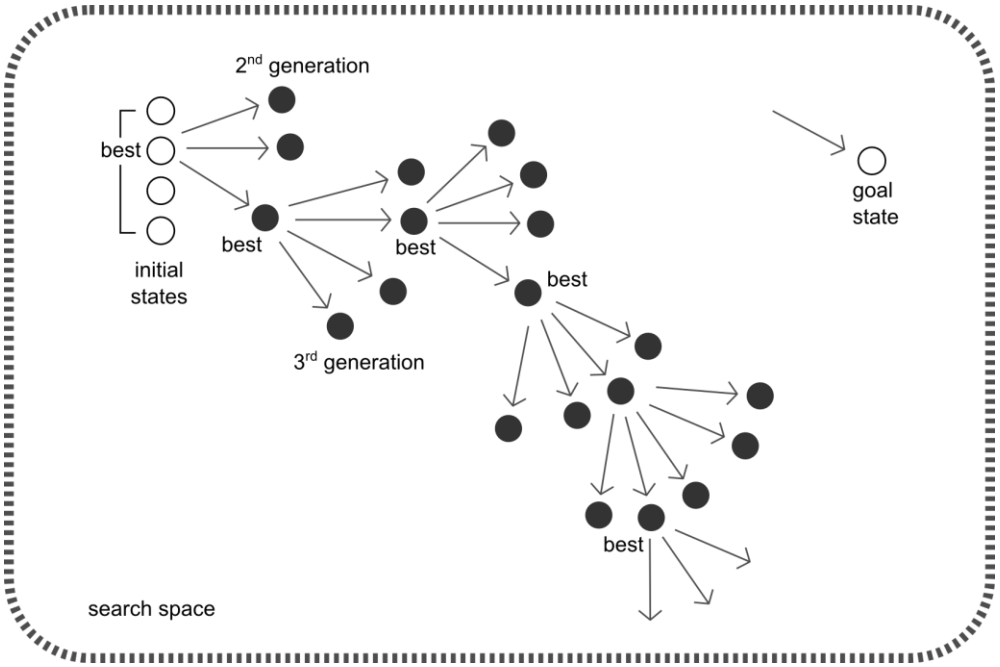


Figure 2.10 : Local search by 'greedy hill-climbing'.

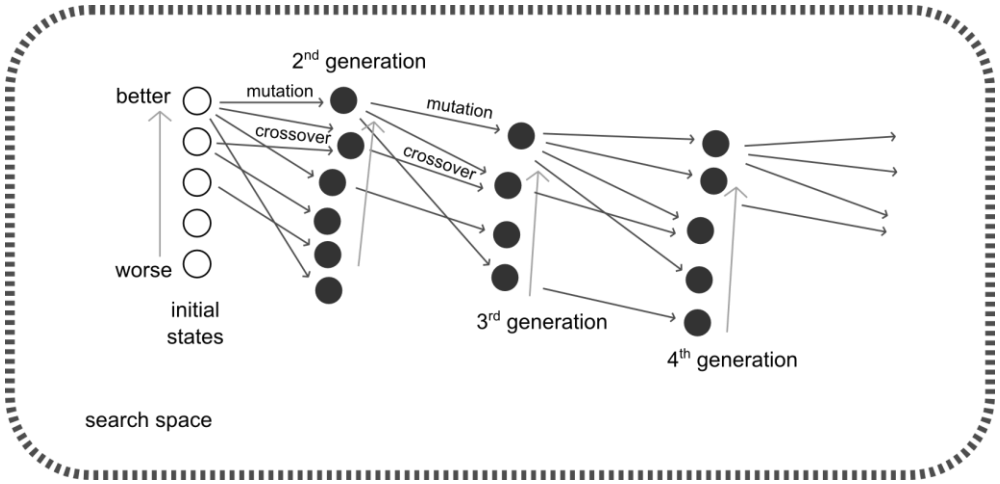


Figure 2.11 : Evolutionary parallel search.

The character of the task environment affects the problem-solving approach adopted to handle a problem. For example, the above-described approaches assume that the environment is fully observable and deterministic; therefore, the problems are tackled by offline algorithms (Figure 2.12). Starting with the knowledge of an initial state, the agent can calculate exactly which state results from any sequence of actions and knows which state it is in (Russel and Norvig, 2010, p. 133). There is no need for percepts that would provide new information after each action.

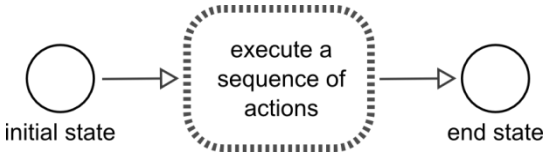


Figure 2.12 : Offline algorithm.

Contrarily, when the task environment is partially observable or nondeterministic, perceiving new information becomes useful. In a partially observable environment, every percept helps narrow down the set of possible states the agent might be in, making it easier for the agent to achieve its goals (Russel and Norvig, 2010, p. 133). When the environment is nondeterministic, percepts tell the agent which of the possible outcomes of its actions has actually occurred. In these cases, the solution to a problem is not a sequence but a 'contingency plan', or a strategy, which specifies what to do depending on received percepts (Russel and Norvig, 2010, p. 133).

This kind of search is performed by online search agents (Figure 2.13). It may be viewed as a process for gathering information about problem structure that will ultimately be valuable in discovering a solution (Simon, 1996, p. 127). In online search, after each action, the agent receives a percept telling it what state it has reached. From this information, it can augment its map of the environment. This way, the agent interleaves computation and action: first, an action is taken, then the environment is re-observed and the next action is computed (Russel and Norvig, 2010, pp. 147-149).

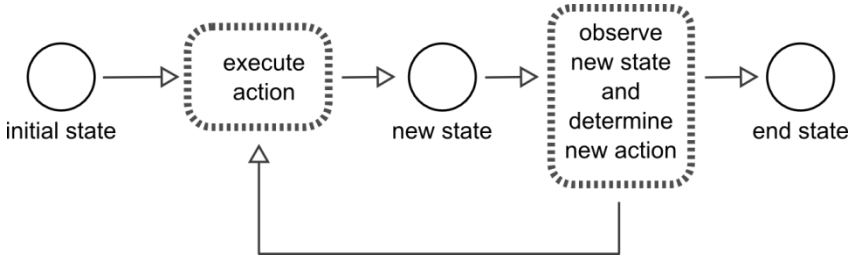


Figure 2.13 : Online algorithm.

Online search is helpful in dynamic and nondeterministic domains and is necessary for unknown environments. In a state of ignorance, the agent faces an exploration problem where it must use its actions as experiments to inquire into the situation, in order to be able to make decisions. Design situations strongly exhibit these characteristics and it can be claimed that design requires an on-line, exploratory attitude. Thus, following the problem-solving approach, design processes have been likened to 'exploratory search'. According to this approach, design is a purposeful

and conscious specification of the actions that must be taken at the present to attain some desired future conditions, where alternative courses of action are hypothesized and their effects are predicted and evaluated against the predefined set of desired conditions (Kalay, 1992). If the evaluation finds that the predicted effects of the actions match the desired conditions, the hypotheses are validated. On contrary situations, the actions must be modified to achieve the effects of the proposed actions. This is done through iterative modification of the proposed actions and the desired conditions until a match is achieved. In this approach, the design process can be said to comprise three major tasks (Kalay, 1992):

1. Defining a set of desired conditions that comprise the objectives to be achieved (Analysis: problem setting).
2. Specifying the actions that, in the opinion of the designer, will achieve the desired objectives (Synthesis: search process).
3. Predicting and evaluating the effects of the specified actions to verify that they are consistent with each other and that they achieve the desired objectives (Evaluation).

Problem-setting phase has to encompass the devising of a proper representation of the state spaces and ensuing operators (or modifiers), which generally requires an adequate abstraction and formalization for the design problem at hand. The desired conditions may take different forms like goals, performance criteria, constraints, or objective functions—fitness functions in the case of evolutionary computation.

However, in order for a design-like task to be handled by a search approach, several difficult problems have to be solved before the process starts. Defining desired conditions as well as the means to test these conditions, or specifying modificatory operators all present difficult problems because of the multi-leveled and underdefined character of design problems. There are several assumptions that have been frequently employed by the design-by-search approaches. In the simplest case, the task environment is taken to be static and deterministic, and the initial state is assumed to be known. Solution procedures cannot handle any unexpected events and are executed without paying attention to the new knowledge that arises during search.

In any real world design task, human designers appear to overcome all the above-mentioned problems. However, most of the times, they do not carry out these tasks in a rigorous fashion and most of their decisions are not stated explicitly. It should be noted that, in more engineering oriented design fields, well-defined situations appear and optimization methods are applicable in these cases. This fact apparently led Simon (1973) to assume that such approaches were ultimately applicable to all design tasks. However, in creative design fields, design research has not been able to extract standard strategies or well-defined objectives, which renders optimization unsuitable.

The above-mentioned explicitness is a requirement for simple problem-solving approaches. According to Simon (1996, p. 132), problem-solving can be viewed as a mere change in representation, making evident what was previously true but obscure; like mathematical derivation. The issue of representation becomes central in this approach and solving a problem simply means representing it to make the solution transparent. Therefore, formalization of a design situation has been seen as a potential approach to convert design tasks into well-defined problems that would be suitable for search or optimization methods. A rigorous formal definition of a design task would provide a suitable representation, though by risking over-simplification.

Informed search algorithms appear to present a way to employ domain specific knowledge during solution processes. This knowledge is crucial in solving complicated tasks and there are several approaches to utilize it. One such approach is case-based reasoning. In case-based design, the process starts with a search for a previous solution (or case) to a similar problem and a solution is generated by an adaptation of the previous solution to the current problem (Kalay, 2004, pp. 261-267). A case typically consists of a problem definition, a solution, and an evaluation of the outcome. From these three components, indexes are generated in order to be used in later searches for a best matching case (Kalay, 2004, p. 262). When a matching case is found, computational systems attempt to assist the human designer in the adaptation process in an interactive session. An exemplary case-based assistant is included in the SEED project (Flemming and Woodbury, 1995; Rivard and Fenves, 2000).

Another approach that attempts to codify and apply expert knowledge in a field is 'expert systems' (Russel and Norvig, 2010). Expert systems try to capture the

knowledge in the form of if-then rules, such as "if the living room has a south facing wall, and if the site is on the northern hemisphere, allow large windows". Such rules have been used in daily practice of design, though it is hard to claim that there is a fixed, systematic, complete, or consistent collection.

From the viewpoint of algorithms, condition and consequent sides of the rules are taken as atomic, therefore their content is arbitrary in the sense that their truth depends on their sources, while the inference is strictly logical. Chaining these rules by way of forward or backward chaining, appropriate results for a given problem and simultaneously the reasons why such a result ensues can be obtained. The knowledge base of expert systems are separated from their inference engine, therefore they are hoped to be more flexible than the hard-coded procedural applications. PREDIKT (Oxman and Gero, 1987) is an experimental system that exemplifies this approach in preliminary design of domestic kitchens.

Another popular rule-based approach is 'shape grammars'. A shape grammar comprises a set of rules that can be applied consecutively to a geometrical construct for modifying it through geometrical transformations (Stiny, 2006). Like expert systems, shape grammar rules include a left-hand side (the condition) and a right hand side (consequent). The main difference of these rules with the rules of expert systems is that the inference results in geometrical constructs. When the part of the current shape matches the condition part of a rule, it can be substituted by the right-hand shape.

2.4 Challenges of Design vs. Human Strategies

In the previous sections, design has been characterized in its general progression and an overview has been given for Computational Design studies, with a focus on the design-by-search approaches. In the following sections, human design exploration will be revisited in order to give a more detailed account of the main difficulties facing Computational Design studies. This investigation will largely follow the 'Schönian' reflective practice perspective and will be mostly congruent with Dreyfus's (1972; 1999; Dreyfus, Dreyfus, and Athanasiou, 1986) influent criticisms

towards AI[8]. Although this investigation will not yield directly applicable recipes, being aware of basic difficulties appears necessary in order to be able to assess a computational approach's potentials and limitations, which is amongst the aims of this thesis study.

2.4.1 Design by search and on-line exploration

As stated earlier, in analyzing design cognition, it has been widespread to use the language and concepts from cognitive studies of problem-solving behavior. However, early hopes for describing design essentially as problem-solving were in the end not justified. Design may involve periods of activity in which problems are solved and the process during those periods may indeed be seen as problem-solving. However, design involves more of the above-mentioned open problems and it is not regular problem-solving. There is no way in which all of design can be reduced to problem-solving activity, or with a better expression, to a series of well-defined problems (Cross, 2006, p. 77; Dorst, 2006, p. 35; Lawson and Dorst, 2009; Lawson, 2005, pp. 31-32; Lawson, 2004, p. 20). Therefore, AI techniques such as heuristic search or logical inference, which require explicit problem structure, are not, on their own, sufficient for developing artificial design agents.

Since design situations are not directly formalizable in terms of simple search procedures, and because existing AI techniques mostly depend on variants of search, for the application of these techniques in design, it appears necessary to devise mechanisms that would be located at an intermediary position between human treatment of design situations and search mechanisms. In response to this requirement, design has been characterized as a two-level process of exploration, which could include search episodes on the lower level. In this conception, search is a process for finding values of variables in a defined state space, whilst exploration is a process for producing these defined state spaces (Gero, 1994). Exploration precedes search and it converts one formulation of the design problem into another. As such, it creates new state spaces or modifies existing state spaces (Gero, 1994).

Gero (1994) further suggested that Evolutionary Computation was capable of supporting exploration in design. This claim has been put to test by Maher, Poon, and

[8] These criticisms will be addressed in the following sections.

Boulanger (1996). Indeed, this was the underlying idea behind their problem – solution co-evolution model that was mentioned earlier (Figure 2.3). We will revisit and discuss this study in Chapter 3, within the context of evolutionary computation.

This picture of a multi-layered process of simultaneous construction of problems and solutions for navigating a vast search space apparently corresponds to the human designers' attitude. However, it is not a trivial task to adapt such a scheme to describe computational processes for design. Available techniques allow only offline agents that give way to rather shortsighted search processes within encapsulated problem definitions. Offline procedures, by definition, do not have means to respond to the multi-level transformations of a design situation[9]. The question then appears as how to incorporate a kind of online agent that is able to appreciate this multifaceted and multi-level process in order to govern its transformations with a strategic view.

Because the task obviously requires a high level of intelligence, a straightforward proposal is to incorporate human agents, which would assume the role of the online intelligent agent. During her exploration within a design situation, whenever needs and possibilities overlap, a human agent may assign a series of tasks to her tools or assistants. This conception is in line with the traditional view of how a human agent and a computational system should collaborate. The human agent would employ all her intuitive and holistic thinking and the computer would carry out repetitive and mechanical tasks in which it excels. It appears plausible to interpret current CAD tools in accordance with this picture. However, when the human agents are so involved within this process, it is not initially obvious solely by this description, whether computational techniques would be helpful in this process; this would depend on the type of task and the specific design process[10]. Moreover, an approach that would try to fit design into a scheme just because the computational system requires it that way would most probably fail in practice. If it is not able to carry out autonomous tasks, a computational system should either enhance the capabilities of the human designer or it should enhance the ease and pleasure of the design process.

---

[9] For a lengthy criticism of offline approaches to design, see Gedenryd, 1998.
[10] For an early but still valid overview of this issue, see Cross, 1977.

The second option involves automated computational systems, which would be able to respond to the design situation as it is being revealed. The first requirement for such a system is an online agent architecture. To illustrate the basic requirements of such an architecture, the 'Situated Function-Behavior-Structure' (FBS) ontology will be examined.

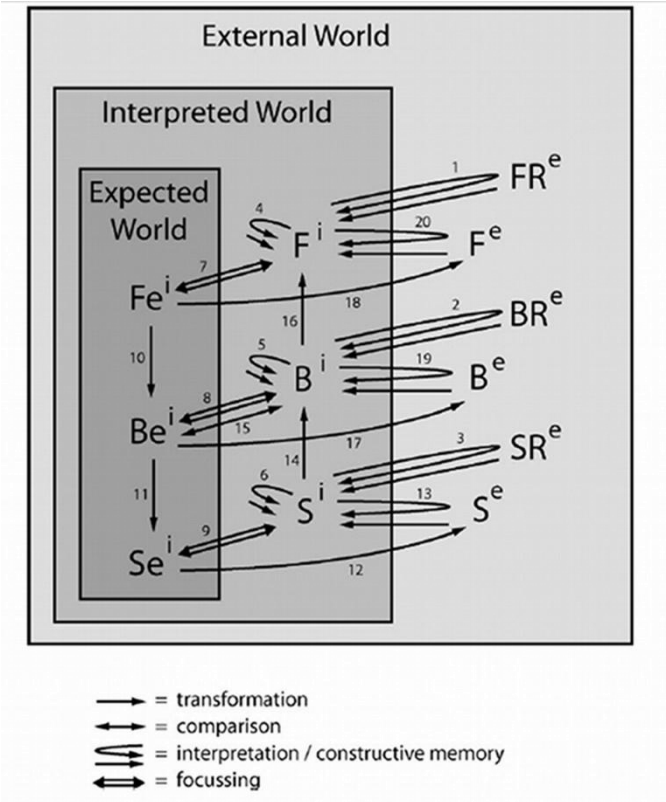


Figure 2.14 : External, interpreted, and expected function-behavior-structure representation of design process (Gero and Kannengeisser, 2007).

The FBS ontology (Figure 2.14) can be seen as a detailed example of how a model can function as a device for understanding and documenting design, while at the same time pursuing practical orientations. It was first adapted to design activity by John Gero (1990; 1998; 2002) and then developed further by Gero and Kannengeisser (2004; 2007; 2008). The FBS ontology simultaneously specifies the entities of a design process and basic actions that push forward design. More importantly, it attempts to do this with regard to the situated character of design actions. The FBS ontology provides three high-level categories for the properties of an object (Gero and Kannengeisser, 2007):

1. The function of an object is “what the object is for”.
2. The behavior of an object is “what the object does”.

3. The structure of an object is “what the object consists of”.

During design activity, designers perform actions that change their environment. By observing and interpreting the results of their actions, they decide on new actions to be executed on the environment, therefore designers’ concepts change according to what they see, which itself is a function of what they do. This interaction between the designer and her environment is an important determinant of the course of design and is called 'situatedness' (Gero and Kannengeisser, 2007). In contrast to the static approaches that attempt to encode all relevant design knowledge prior to its use, situated design uses the knowledge that is grounded in the designers’ interactions with their environment (Gero and Kannengeisser, 2007). Gero and Kannengeisser (2004) have modeled situatedness as the interaction of three worlds:

1. The external world is the world that is composed of representations outside the designer or design system.
2. The interpreted world is the world that consists of the sensory experiences and concepts of the designer or design system.
3. The expected world is the environment in which the designer or design system predicts the effects of actions according to the current goals and interpretations.

These three worlds are linked together by three classes of activities (Gero and Kannengeisser, 2007):

1. Interpretation transforms variables that are sensed in the external world into the interpretations of sensory experiences, percepts, and concepts.
2. Focusing takes some aspects of the interpreted world and uses them as goals for the expected world.
3. Action is an effect that brings a change in the external world.

Locating function, behavior, and structure of a process in each of these worlds results in nine ontological categories, which is given in Table 2.1 (Gero and Kannengeisser, 2007).

Table 2.1 : Ontological categories in situated FBS ontology.

	Function	Behavior	Structure
Ext.	External (F^e)	External (B^e)	External (S^e)
Int.	Interpreted (F^i)	Interpreted (B^i)	Interpreted (S^i)
Exp.	Expected (Fe^i)	Expected (Be^i)	Expected (Se^i)

Additionally, the framework represents external requirements related to the function (FR^e), behavior (BR^e), and structure (SR^e) of processes. Finally, in Figure 2.14, the 20 activities that connect these categories extend the ontology to explore the notion of situated processes (Gero and Kannengeisser, 2007). Gero (1990) proposed eight fundamental steps in design, which were mapped by Gero and Kannengeisser (2004) onto the 20 activities in the situated FBS ontology (Table 2.2, Gero and Kannengeisser, 2007). The numbers of activities correspond to the activities illustrated in Figure 2.14.

Table 2.2 : Mapping of fundamental steps and activities in situated FBS ontology.

	Fundamental steps (Gero, 1990)	Corresponding activities (Gero and Kannengeisser, 2004)
1.	Formulation	Activities 1–10
2.	Synthesis	Activities 11 and 12
3.	Analysis	Activities 13 and 14
4.	Evaluation	Activity 15
5.	Documentation	Activities 12, 17, and 18
6.	Reformulation type 1; redefines the structure state space	Activity 9, with activities 3, 6, and 13 as potential drivers
7.	Reformulation type 2; redefines the behavior state space	Activity 8, with activities 2, 5, 14, and 19 as potential drivers
8.	Reformulation type 3; redefines the function state space	Activity 7, with activities 1, 4, 16, and 20 as potential drivers

FBS ontology can be interpreted to designate an agent architecture for a situated autonomous design agent together with a minimum list of types of design moves. A series of fundamental design tasks (formulation, synthesis, analysis, evaluation,

documentation, and reformulation) are mapped onto a set of activities (transformation, comparison, interpretation, and focusing) that process and transform information between three distinct worlds (external, interpreted, and expected). However, types of moves such as transformation, comparison, interpretation, and focusing, especially between external and interpreted worlds appear as difficult tasks. Obviously, it is not sufficient to provide the architecture; it is also necessary to find ways to carry out the defined design activities. Most of these activities require high-level mental capabilities for which no automated technologies exist yet. The depth and breadth of an exploration that would be carried out with an agent that is based on the FBS framework would be determined by how the external world would be interpreted as the task environment. The type of interpretative skill, which enables exploration and which is observed in humans is still not attained by artificial reasoning systems. Considered in this light, it would be too optimistic to assume that a simple two level co-evolutionary scheme is in a kind of continuity with how humans carry out exploration. This is where there is a leap, rather than continuity.

This is most obvious in a comparison of working styles of artificial systems and human agents. Artificial systems generate an abundance of alternatives within a search path, because even in the case of local search, the generation of new alternatives is mostly random. On the other hand, human designers proceed through a limited number of alternatives, while carefully considering each transformative action and ensuing state from a multitude of aspects. Each move and each evaluation is intuitively rich, though most of the examination is not explicit. Humans explore in a deeper and more comprehensive manner, while artificial systems are relatively shallow interpreters. They have to compensate their shallowness with generating more search points, which easily becomes ineffective when the problem gets more complicated. Nevertheless, whether they are similar to human exploration or not, complex evolutionary systems, through which a process could gain a strategic layering, appear as a potential development path for Computational Design, though with apparent limitations. Complex and hierarchical evolutionary systems will be discussed in Chapter 3, where an example algorithm, the "Interleaved Evolutionary Algorithm" (Interleaved EA) will also be introduced.

2.4.2 Complexity of design situations

In the case of architectural design, the products of a design process are complex entities, involving systems, sub-systems, parts, sections, and functions that amount to an enormous variety of materials, details, and systems. Designing and constructing a building involves such an endeavor that hundreds, sometimes thousands of participants contribute to it and dedicate some of their mental processing capability to a minute detail or to an overall issue. As Jones (1992, p. 41) noted, “the main difficulty in any form of designing is that of coping with the complexity of a huge search space filled with millions of alternative combinations of possible sub-components”. This fact will be referred to as the "inherent complexity" of design.

Yet, there is another kind of complexity, which concerns the negotiated aspects of design. At any design situation there will always be competing viewpoints, which have to be settled through negotiation. This issue was amongst the reasons why design has been characterized as "wicked" (Rittel and Webber, 1973; Buchanan, 1992). This type of complexity is related to the essentially undefined aspects of design, because it only appears within a particular situation and it can only be solved by dispute or negotiation. Therefore, this property of design situations will be referred to as the "inherent undefinedness" of design. A diagram of design situations is given in Figure 2.15, which maps the basic issues that contribute to both types of complexity. Throughout a design process, components of the co-evolving problems and solutions generate an inherent complexity, while a multitude of agents continuously bring forward differing viewpoints and framings, exacerbating the inherent undefinedness.

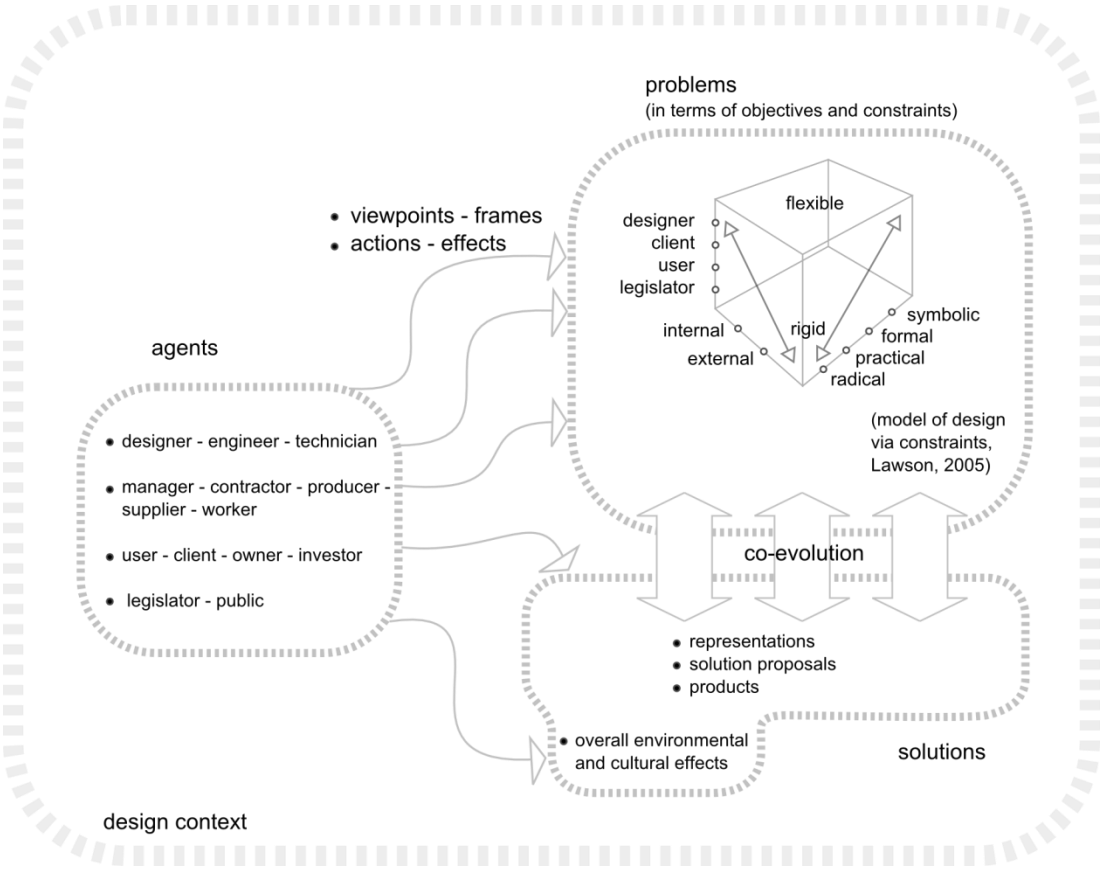


Figure 2.15 : Basic constituents of a design situation.

These two types of complexity of the design problem should be considered as the basic problems for search-based and formal approaches. Any formalization attempt will face insurmountable hardships in initial problem formulation (because of the inherent undefinedness), and any one-level brute force or heuristic generate-and-test approach will encounter combinatorial explosion (because of the inherent complexity). In combinatorial problems, the search space grows very rapidly because of the combinatorial factors. From the aspect of objectives, each design situation involves several constraints, objectives, and performance criteria some of which bear a normative and subjective character that should be decided through forms of social negotiation. Even worse, these objectives often impose conflicting requirements.

To overcome these difficulties, reductive representations have been frequently utilized in computational approaches, which narrow down the search space by converting and confining problems into well-defined borders. However, due to the large number of important variables, even after this reductive definition, computational complexity of a design task will remain high. Decomposition of a task is a potential remedy in complex problems against the problem of combinatorial

explosion. However, this is not straightforward in most design tasks where sub-tasks are usually dependent.

On the other hand, human designers tend to inquire into a design situation through solutions, with the help of prefabricated strategies and solution patterns. These solutions often exhibit a holistic character, in the sense that they pose integrated solution strategies towards several aspects of the problem in one move. A traditional building element such as a simple brick wall that is pierced through by several apertures is being used in a great majority of buildings, as a solution, which simultaneously concerns appearance, structure, envelope, illumination, ventilation, and insulation. In other words, it is a popular prefabricated strategy[11]. Following Christopher Alexander (1977), these types of integrated prefabricated solutions may be called "patterns".

A solution-based approach enables the designer to put forward a solution path by providing probable solutions for a variety of aspects of the problem all at once. Moreover, having once decided on a solution path, human designers temporarily become indifferent to other possible alternatives. What has transpired is the application of a simplifying assumption that makes the decision process tractable (Rowe, 1987, p. 54). This is not simply a narrowing down of a search space; rather it is the beginning of the structuring of search spaces, hence a positive act. This structuring is carried out through a flexibility that can undo any decisions or modify them on the run, which appears as important as the solution-based strategy. A mere solution-based approach would not yield the desired results without the dynamic reappreciation of renewed situations.

The main difficulty for computational approaches is the dynamism and flexibility in the application of these strategies. When human designers make strategic decisions, they seldom, if ever, systematically evaluate the outcomes of all possible decision sequences. Rather they choose combinations that are habitual, that seem to offer satisfactory results, or perhaps seem to involve the least amount of risk (Rowe, 1987, p. 54). Although, in a sense, prefabricated, these strategies are not strictly determined in a problem-setting phase to be applied in a rigid manner. On the contrary, human

---

[11] The term "prefabricated strategy" is borrowed from Jones (1992).

agents work in an online fashion during the process, changing and modifying their strategies and evaluative criteria on the run, by using the feedback gained at each moment. Such feedback not only concerns the limited scope of the problem area, but a broader field concerning all participants' dynamically unfolding wishes and sometimes cultural and personal issues. In brief, design requires a combined use of both flexible prefabricated strategies (instead of rigorous problem formalizations) and a high level of intelligence, to be applied dynamically for providing both rationale and contextual evaluations throughout the design process. A combination of these enables a dynamic and situated design process.

2.4.3 Ubiquity of interpretative skills

When there are the means for a continuous re-evaluation of a complex situation and an additional ability to re-define a problem at each new juncture, then the degree of success expected from each applied strategy may diminish, and proposals may truly become provisional or tentative. A series of actions might even be chosen because they are not apparently useless and the designer needs some concrete direction to move on, i.e., just because they enable moving on and creating further possibilities. This can also explain why some acts of human designers appear frivolous and gratuitous. In some of these cases, there might not be an unrevealed deep structure, but just frivolity, which may still have utility. Thanks to her general intelligence and dynamic evaluation capabilities, the designer is freed from the impossible need to calculate every outcome before moving on; she acts, collects the outcome, re-evaluates the situation, and modifies her plans. This also enables her to consider newly revealed information. This is, in a way, continuously scanning a region of potentialities.

Even with well-defined problems of optimization, multi-objective evaluation poses important difficulties. A high number of objectives have to be satisfied for architectural problems. These include formal, aesthetic, functional, structural issues as well as market value, environmental quality, social influences, and relationships with an urban context. The manner of evaluation is important in coping with such a complex problem; when, in which particular situation, and how (deep, shallow, particular, detailed, general, overall, ethical, aesthetic, functional, etc.). The answers to these questions will strongly depend on particular contexts. The designers have to

employ their dynamism and flexibility in the evaluative actions. One of the skills that a designer must have is to be able to suspend judgment to allow the preliminary ideas to mature before they are subjected to harsh criticism. Knowing when and how to evaluate, as an individual and in teams is a core design skill (Lawson, 2004, p. 299).

2.4.4 Problem of (re)framing and an expertise built over everyday intelligence

The process of reflection-in-action and especially the particular version that Donald Schön (1983) calls "reflective conversation with the materials of a situation", is an essential part of the artistry with which practitioners cope with uncertainty, uniqueness, and value-conflict in all domains of professional practice (Schön, 1984). According to Schön, designers operate within seeing-moving-seeing cycles, during which they constantly re-evaluate the situation. The reflection-in-action model has been brought forward by Schön (1983) as an alternative to the technical rationality, which resulted in the problem-solving view of design. In the problem-solving approach, the problem is abstracted from irrelevant issues and the task environment is rather static, even for the online agents. Contrarily, in the reflection-in-action view, the essential issue is the overall dynamism regarding the whole situation. It is apparent that there is not much to formalize in a design situation as in chess or mathematics, where the basic units are discrete and well-defined entities. These latter problems operate on a separate universe of their own, which is by no means the situation in architectural design.

The online interactive mode of the design process is not accidental (Gedenryd, 1998); on the contrary, it stems from the requirements of the design situation, which involves many undefined aspects. First, the information regarding the particular design situation is mostly incomplete. Secondly, general strategies and rules regarding design products and designing is lacking. Thus, the design agent has to be able to operate with limited information and knowledge, and has to use strategies to acquire and use new information. This is another reason why the design agent should have general intelligence, in addition to expertise, which is useless without intelligence.

2.4.5 Exploratory journey with the help of mental maps

Design becomes an exploratory journey with the help of mental maps. Designers are creative individuals and a core skill for creative design is to possess guiding

principles and strategies to structure vast search spaces while travelling through them. In this exploratory process, carrying and utilizing a broad and flexible basis of both everyday and domain-specific knowledge is required. The knowledge is stored in the form of associative maps, and an additional ability is required to continuously renew and interpret these maps (Boden, 2004).

The importance of knowledge for design should be emphasized. Design activity requires several types of knowledge. Expertise in a specific field of design is built over basic mental capabilities and it comprises the above-mentioned mental maps in addition to heuristics and tacit skills. It is also important how these types of knowledge are gained, stored, retrieved, and used.

In design, knowledge is utilized on several levels. One level concerns the strategies and guiding principles to structure a design situation and move forward, another regards expert evaluation, which is in turn dependent on commonsense knowledge. On a separate level, designers seem to develop their own programme of intellectual endeavor (Lawson, 2004, p. 300), which can be seen as design philosophies or sets of values on what designers hold as important in their own domains. These can be fairly varied, yet are important in guiding a designer.

Designers use each project as a way of researching their chosen area, improving their understanding of it and at the same time developing their guiding principles (Lawson, 2004, p. 300). Thus, an important source of knowledge is experience of past practices; while another is examining and experiencing design products designed by others. These are stored and retrieved as precedents. Learning from their experiences and from past examples throughout their careers, designers build a formal and strategic repertoire, not in a simple cumulative manner but by continuously updating their associative maps.

Designers keep few rules that tell them how to get from problem to solution, but rather they have a great deal of knowledge about existing solutions and their potential affordances (Lawson, 2004, p. 300). It is not easy to discriminate the procedural knowledge or strategies of design from the knowledge of cases; these form a unified repository, which designers are able to make use of in their solution-focused strategies.

2.4.6 Design knowledge and rigor: Design Methods Movement

Human beings are natively equipped with basic capabilities and a facility for understanding and acting inside the world and a designer's education starts with birth. However, instead of explicit rules and strategies, a referential usage of knowledge—in the form of precedents or references—is frequently observed in design. Beside patterns, precedents, and typologies, several other terms have been proposed for similar solution-based or procedural mechanisms such as cases, prototypes (Gero, 1990), seeds (Frazer, 1995), schemata (Lawson, 2004b), gambits (Lawson, 2004b), style (Simon, 1975), and preconceptions (Janssen, 2004). Before being able to design novel products, a designer has to gain a considerable expertise in the form of both precedents and skills.

The precedents possibly address several issues simultaneously in an integrated manner and it is up to the designer to choose which precedent, for which aspect, and in which context. The precedents may reside on different scales and levels—from a building typology to a window detail—and have an ability to affect other scales and levels in a highly flexible manner. For instance, typologies embody principles that designers consider unvarying and they allow designers to apply knowledge about past solutions to related architectural problems (Rowe, 1987, p. 85).

A basic requirement for the utilization of precedents is recognizing features of situations that make connections with apparently remote sets of ideas. Recognition of the connections between a precedent and a situation is essential for making links between problems and solutions (Lawson, 2004, p. 301).

A fundamental issue regarding design knowledge is that a complete listing or classification of this kind of knowledge appears out of the question. The dynamic nature of the issue makes it hard, if not impossible, to write down a comprehensive and stable list of possible actions and effective strategies. Each tried and known strategy is only a tentative proposal and should be modified in each new context; it is only a starting point and each context demands a unique combination of strategies some of which would be novel. Apparently, some sort of strategic knowledge has to be posited in order to explain design expertise. However, each application of a principle modifies it to adapt to a new context and with the changing context, new heuristics, principles, and references are built and added to the knowledge structure.

A type of high-level general intelligence is required as a basis for this dynamic operation. For these reasons, it can be claimed that design is better off without rigorous methods. Design does not require rigorous strategies; rather, it requires general intelligence. The brief history of the Design Methods Movement supports this claim.

Proposals for new design methods began to appear in the early 1960's, mostly following the developments in operational research that originated in the Second World War (Cross, 2006). Aspirations to convert design into a scientific field based on the model of the technical rationality quickly launched design methodology as a field of inquiry. The desire of the new movement was to base the design process, as well as the products of design, on objectivity and rationality (Cross, 2006, p. 119). Researchers in the field were searching for rational methods for incorporating scientific techniques and knowledge into the design process to render design decisions rational (Bayazit, 2004). A rational, systematic methodology would help establish design as a scientific field. At the same time, with methods that are more rigorous, it was hoped that design education and design processes would become more effective and the quality of design products would improve (Cross, 1977).

A series of analysis – synthesis – evaluation models and step-by-step design procedures were prevalent in these early years (Cross, 1977), as was discussed in previous sections. A characteristic of the era was an interest in the externalization of design knowledge by means of charts, diagrams, lists, specifications, and so on (Cross, 1977; Broadbendt, 1973).

From the very beginning, design methods researchers were interested in computational methods. There seems to be a correspondence between the design methods movement and computational techniques. Furthermore, if design could have been encapsulated into step-by-step procedure definitions, or better, into formal systems so that they could be "computable", the problem-solving techniques of the era would allow an automation of design processes. The decade culminated with Herbert Simon's outline of the "sciences of the artificial" (1996, first published in 1969) and his specific plea for the development of a science of design in the universities: a body of intellectually tough, analytic, partly formalizable, partly empirical, and teachable doctrines about the design process (Cross, 2006, p. 120).

However, during the 1970's there emerged a backlash against design methodology and a rejection of its underlying values, notably by some of the early pioneers of the movement (Cross, 2006, p. 120), like Christopher Alexander (1984, originally published in 1971) and John Christopher Jones (1984, originally published in 1977). The proposed methods started to seem too systematized for design and a waning of enthusiasm was being observed (Broadbendt, 1968, cited in Cross, 1977).

Indeed, at the beginning, systematic design was aimed to be a synthesis of intuition and rigorous mathematical or logical treatment (Cross, 1977, p. 12). However, in the end the field was leaning more towards rigorous schemes, which started to seem incompatible with real design processes. Consequently, it became necessary to acknowledge that there had been a lack of success in the application of scientific methods to everyday design practice.

Fundamental issues have also been raised by Rittel and Webber (1973) who characterized design and planning problems as essentially incongruous with the techniques of science and engineering. Although design methodology continued to develop strongly in engineering oriented fields and in some branches of industrial design (Cross, 2000; 2006, p. 120; Dorst, 2006), design research in fields like architectural design largely abandoned a search for rigorous methodology, mostly in favor of researching the behavior of human designers.

The tacit nature of design knowledge is largely acknowledged today (Niedderer, 2007). Explicit, propositional knowledge does not abound in design fields. Most of the knowledge is procedural. It is learned through designing, stored in a tacit manner, and transferred via practice, during design assignments and through discussions on design cases. Expert designers exhibit a higher level of skill compared to the novices, but it is hard to explicate this in the form of a consistent collection of rules or propositions. Instead, broad outlines can be presented, which, however, does not help much in particular design situations. This is why design has no textbook. Nevertheless, these broad outlines might help in understanding the hardships encountered by design computation.

2.4.7 Dynamic focus and continuous (re)framing

A family of design strategies has been about structuring the design problem. From the perspective of search, this amounts to a narrowing down of the search space.

Starting from and proceeding through a small number of solutions and following a minimal number of threads is one such strategy (Rowe, 1987, p. 68). This strategy can also be seen as a kind of selective focusing.

A designer's starting point is usually a single design which she can visualize (Jones, 1992, pp. 22-23). The traditional design method is to draw and redraw successive alterations on layers of tracing papers, gradually elaborating over—and diverting from—an initial sketch. Human designers tend not to produce many alternatives; instead, they conduct their inquiry over mostly a few alternatives at each juncture. This can be seen as a constructive process, a simultaneous development on several layers and scales that loosely correspond to design tasks and objectives. Each inquiry layer starts at one time during the process and continues intermittently until the end of the process; as constantly informed by the developments in the other layers.

The generation of only a small number of alternatives requires intelligent and dynamic generation and evaluation procedures; the designer has to know which alternative to generate and how. After generating a hopefully high-rated candidate solution, she has to be able to understand the essential issues at that precise point, choose the right type of evaluation approaches, and conduct these with regard to the specific context.

Designers exploit another strategy at this point. The context is mostly understood through a specific framing, which further reduces the possible ways the undefined cultural and aesthetic issues could be understood. A particular framing gives structure to the problems. Without any framing, every act would be equally gratuitous, criteria would vanish, and problems would evaporate.

Remember the library example, our architect has chosen the entrance canopy as the primary element for an expression of style and at the same time as a social space, hence a central motif in her design. Assume that she has seen this motif in a magazine. Now in a specific context, she remembers that canopy and matches the conditions of her situation with the precedent’s. Her main aim is to design an impressive facade. This has been her earlier framing, and now she finds out that by the implementation of this canopy, she can simultaneously convert the entrance space to a kind of social space, which appears to her as a charming idea. Now her framing is updated and she sees the situation with respect to the interest of the public.

This new frame is stronger than the earlier personal perspective. However, when the costs are negotiated, she faces the drastic situation that there would not be enough finances to apply the canopy idea. She does not bring forward a series of alternative ideas. Instead, she tries to find ways to retain her framing before moving on to a new solution, but in the end, she modifies her frame again into a rather functional one. If we consider each framing, each brings forward another series of criteria and deemphasizes others. If the main idea is a personal distinguishment as an architect, a cheaper stylistic invention may satisfy the main criterion and the possibilities are only limited by the designer's abilities. If the idea is creating a social space, then, thinking that her arguments are strong and the element is important, the designer may fixate on the element, so that the branching factor at this level is reduced to one. If the issue is purely functional, several existing shading elements can be compared in terms of prices, appearances, and functionality and the problem becomes better defined.

An important requirement in the framing process is the motivations brought by the participants of design processes. While the clients and other parties bring a series of objectives, desires, and criteria, these often leave the design problem undetermined or underdetermined (Dorst, 2006; Lawson and Dorst, 2009). Therefore, designers often bring additional motivations and self-imposed constraints into the situations as was illustrated with the library problem, where the architect was motivated by her personal aspirations for her practice, by preconceptions about architecture, and by stylistic issues. This reveals an important limitation on the side of the artificial agents, which have to struggle through enormous unstructured search spaces in a rather unguided manner.

Style has an important role in the framing action. It may stem just as much from the decisions about the design process as from alternative emphases on the goals to be realized through the final design. It can be related to the manner of conducting a task, as Simon (1975; 1996, p. 130) proposed. An architect who designs buildings from the outside in will arrive at quite different buildings compared to one who designs from the inside out, even though they could agree on the characteristics that a satisfactory building should possess. Style may concern visual expressions or functional strategies; either way, it is what the designers construct throughout their careers and which provides the paths to personal framing and structuring actions.

2.4.8 Integral approaches

Through the act of framing, a problem is tentatively structured so that hypothetical criteria and expectations can be developed for interim evaluations. Because a minimum number of evaluation criteria and alternative proposals are determined at each juncture—and in a hierarchical fashion (there are at any point both essential and inessential criteria)—the problems with combinatorial explosion can be overcome.

In a single design scenario, at some certain stage, and with a certain viewpoint—as in the library problem—there are just a few alternative solutions to be evaluated in a multifaceted and deep manner. Moreover, these evaluations may be carried out with mostly relaxed success thresholds. Generally, no optimum is sought for; the proposal has to appear merely viable. Even if some proposal appears improbable, designers might tend to suspend judgment and retain the proposal to see whether a future condition or idea might change the situation so that the proposal appears viable. At any stage, the designer may keep several incompatible frames simultaneously, as alternative scenarios.

The requirement for a small number of threads or alternatives at each juncture is another reason why integrated solutions are a necessity. As with the example of the brick wall, a unified, integrated, holistic proposal enables the designer to propose at one move well-tested answers to a multitude of problems. The concept of ‘affordances’ may be used to depict this phenomenon (Lawson and Dorst, 2009, p. 198). Each path, each proposal offers potentials as well as limitations.

An alternative strategy could be to decompose the problems in terms of basic requirements. The idea is first, to discover individual requirements and cluster these atomic elements according to their interrelations, then, to devise integrated solutions for each of the clusters, and finally, to bring together these solutions as an answer to the whole problem. This approach has been proposed by Alexander in his work on an Indian village (Alexander, 1964). However, there are very few applications of the approach, and Alexander himself abandoned it in favor of the idea of a "Pattern Language", which readily starts with integral strategies, on a higher level than the basic requirements. One problem with the approach is the difficulty of decomposing design situations, which exhibit highly interrelated requirements. Moreover, there are a huge number of these requirements and determining the relationships between them

is a tedious task. Even if this was accomplished, it is not a straightforward task to integrate partial solutions to an integral whole.

On the other hand, an integral approach like a design pattern enables the designer to start with the whole on an abstract level and detail a series of layers and aspects during the design process whenever necessary. Using previously tested elements and approaches, which are also integral strategies, eases the burden of reconciling complicated issues. This is also how design knowledge in the form of precedents and prefabricated strategies are put to use.

2.4.9 Dynamic decomposition and integration

Interrelated issues result in combinatorial and sometimes intractable problems, which is mostly the situation in design. The concept of hierarchical decomposition is thought as a remedy to this situation, as illustrated by Alexander's above-mentioned study. If a suitable manner of decomposition were found for a complex problem, then the problem would become tractable:

> The basic idea is that the several components in any complex system will perform particular subfunctions that contribute to the overall function. Just as the 'inner environment' of the whole system may be defined by describing its functions, without detailed specification of its mechanisms, so the 'inner environment' of each of the subsystems may be defined by describing the functions of that subsystem, without detailed specification of its submechanisms. To design such a complex structure, one powerful technique is to discover viable ways of decomposing it into semi-independent components corresponding to its many functional parts. The design of each component can then be carried out with some degree of independence of the design of others, since each will affect the others largely through its function and independently of the details of the mechanisms that accomplish the function (Simon, 1996, p. 128).

For the cases where a complete decomposition is not possible because of the dependencies between the terms, Simon (1996) brought forward the concept of "near-decomposability". However, many design problems are difficult or impossible to decompose without prejudice to performance, cost, weight, appearance, or other objectives (Jones, 1992, p. 51).

There might be several ways to decompose a design problem. One is according to requirements and corresponding parts of the products as illustrated above with Alexander's approach. Another approach might be decomposing the problem in

terms of processes. A complex process might be decomposed into sub-processes, which are possibly operating simultaneously on different layers and scales. Thus simplified, the sub-problems could become tractable. Indeed, both strategies are regularly practiced by human designers.

Designers approach a problem from different viewpoints, not only in terms of requirements, objectives, and constraints, but also in terms of scales and levels. This helps them to frame and target different components of the situation in a selective manner. This means that the very act of framing is a decomposition strategy. Design problems are decomposed in terms of both process and objectives.

The structuring of a design process into phases is a type of problem decomposition through process. Although they exhibit a complex circularity, design processes are observed to proceed from a rather general and abstract level towards detailed specifications, through a series of phases that are delimited by deadlines, such as a meeting with a decision-maker. A series of production tasks may be defined for each of these thresholds. These phases may take forms like preliminary design phase, production of scale drawings, detail drawings, or production drawings.

In design, the task is never confronted in its totality. Instead, abstractions are made regarding the specific phase, scale, and level. After constructing an abstracted and simple version of the problem, some aspects of it are chosen to be tackled. The selection of which problem and which aspect to tackle is a matter of dynamic focusing, which diminishes the complexity of the issue at hand:

> . . . We can see that the traditional way of dealing with complexity is to operate at any one time, only upon a single conception of the whole. This, as embodied in a scale drawing, is a means of drastically reducing what would otherwise be an unmanageable number of decisions to be taken in fixing the shape and position of each part of the design (Jones, 1992, p. 30).

Referring to an empirical study, Akin (2001) reports to have found that, human designers mostly reconcile issues, by focusing on a couple of them at a certain time, and by visiting these couplets in a serial manner. The main requirement for this process is not a greater searching capability, but a more developed interpretative intelligence, which should be able to shift its focus towards the essential items as they appear.

2.4.10 Different kinds of performances

The operation of evaluation is directly linked with the concept of performance in design. Consider the example of facade systems. Facades fulfill multiple functions within a building. They enable day lighting for illumination and admit direct sunlight to help heating. However, they carry out these tasks with the expense of heat and noise insulation. These types of performance requirements can be measured with specific methods; hence, performance is often understood in terms of numerical specifications.

On a broader level, performance requirements for building systems include their ecological footprints, which concern energy and resource usage, as well as toxicity. There are approaches for the assessment of these qualities; however, this might not be straightforward due to the ethical aspects of the issue.

It is reasonable to think that, a good building is one, which performs well in terms of comfort, building health, energy performance, ecology, and economy. The kinds of performances that are mentioned until this point are mostly measurable. From a practical viewpoint, a performance measure enables a simple comparison between different alternatives, where it is rational to choose the better performing alternative.

The situation becomes more complicated when there are more than one performance requirements. Because each measurement resides on a different scale, there is usually no straightforward way to compare one performance type with another. For example, there is no obvious way to compare illumination with noise insulation. Therefore finding an optimal balance for all the requirements is a difficult problem.

There are other aspects of a building that strongly influence the appreciation of it. Kinds of disputed performances might be encountered in political, cultural, aesthetic, or ethical contexts. For example, facades determine the appearance of buildings. The style of a facade may have influences over human psychology or over a specific urban context. Appearance of a building may have a communicative function; it may exhibit the status or institutional identity of its owner, or the personal style of its architect. These latter aspects should also be considered amongst the types of performances of a building. However, there are several problems that prevent a measurement of these aspects. First, issues such as psychology and culture are essentially complicated and their mechanisms are not easy to explicate. Secondly,

even if these mechanisms were understood, issues like taste are essentially subjective. Every individual possesses a unique combination of tastes. Thirdly, actors in the real world exhibit different and sometimes competing interests. In these cases, the decisions are inevitably subject to negotiation.

Concerning aesthetic appreciation, there has always been an interest in finding universal aesthetic principles, e.g. the golden ratio. However, it is hard to claim that this has been achieved. Instead, aesthetic appreciation appears to be affected by many interrelated factors and it is highly subjective. Because such issues are highly subjective, and because there is no universally valid position regarding them, an agent that deals with these issues should be equipped with an individual stance regarding each of these issues, to be able to enter into a negotiation with the situation. In other words, agents should have characters. The painting system Aaron is an interesting example for such an artificial agent with its own style (Url-15). Aaron has been developed by Harold Cohen over several decades and is a successful artificial painter, capable of producing figurative paintings within the same visual style; it has an artistic character.

Although style has been an important determinant of aesthetic appreciation, it is not easy to describe it, as it appears as an umbrella term for several distinct mechanisms. Nevertheless, there have been many studies, which managed to codify a specific formal style. In a seminal study, Stiny and Mitchell (1978) developed a shape grammar, which was able to generate Palladian style villas. Duarte (2001) developed a shape grammar system, which can generate houses based on the manner of Alvaro Siza's Malagueira houses. Likewise, Çolakoğlu (2005) developed a shape grammar for traditional Balkan houses. In the generative side, Eiben, Nabuurs and Booij (2002) used Evolutionary Computation to imitate the styles of Mondrian and Escher.

In these examples, style is the by-product of a production procedure. In a similar manner, style may stem from algorithmic procedures as observed in generative art and design, where the human agent and her code constitute a symbiotic complex to generate a distinctive stylistic expression. In most generative approaches, human agents do not have a total control over the process and the algorithms are responsible for some aspects of resulting visual styles. In these cases, a separate effort for the definition of style is not required, as the procedure is already known.

It should be noted that, despite the existence of the above-mentioned approaches, visual style has traditionally been viewed as a human field. Consequently, Computational Design studies tend to focus on the more easily measured aspects of design, which are seen as easier to automate. Because a design's success is also strongly dependent on undefined aspects, interactive approaches have been developed for additional evaluations by human agents. This way, technical issues are considered side-by-side with the aesthetic issues. However, it is also widespread to simply ignore the aspects that are hard to pin down.

The Gene-Arch system developed by Caldas (2003, 2005, 2006, 2008) exemplifies such a reductive approach. The system follows a performance-based approach for determining room shapes and aperture dimensions, considering psychological or aesthetic issues only in terms of static constraints. It can be claimed that the concept of performance should be understood in a broader sense. With a similar motive, Bitterman (2010) proposed a cognitive approach in order to take into account what he calls "soft aspects" within evolutionary design. He developed a model for human perception, hence tried to transform a soft aspect into a hard-coded static model. This model was then used for the evaluation of visual privacy.

In summary, design processes involve a variety of objectives, requirements, and constraints; including measurable specifications as well as undefined ones. All these objectives constitute a complex and dynamic pattern of goals. Computational Design studies should be able to incorporate all of these types of objectives and corresponding types of performance evaluations. For a more complete range of performance types, a fourfold classification scheme can be proposed as follows[12]:

1. Prudence: Implicit constraints that are usually tacitly expected. For example, a wall would transfer its loads to a basement; a bedroom would be wide enough to include a bed, etc. Some of these constraints can be controlled within parametric definitions, as exemplified by BIM and parametric approaches.

[12] The terms well-defined, underdefined, and undefined are used with reference to Dorst's (2006) threefold classification for design problems, i.e., determined, underdetermined, and undetermined.

2. Well-defined performance: These can be calculated through specific methods, such as simulation methods or objective functions. Examples are structural stability, insulation, illumination, ecological footprint, cost, etc.

3. Underdefined performance:

    3.1. Functional: Such as the quality of a spatial organization, which requires integrated evaluation of various qualities. These qualities are hard to completely define. Several approaches have been developed for the evaluation of spatial organizations, such as adjacency matrices and space syntax methods; however, these are not as multifaceted as a human designer's evaluative capabilities.

    3.2. Visual: Buildings are expected to exhibit a high degree of visual quality. Evaluations regarding these issues exhibit a subjective character and are mostly left to human designers.

4. Undefined performance: The above-mentioned types of performance are the ordinary requirements for a building. However, because of the varying viewpoints and unique conceptual frames that are developed through any design process, other requirements, constraints, and objectives are continuously generated and negotiated. These tend to take the form of tentative agreements and are usually peculiar to a specific design process.

The first two headings concern the types of performances that are rather straightforward to implement. The requirements that are grouped under the third heading are mostly problematic for computational approaches, and are usually carried out by human agents. The main difficulty of the requirements under the fourth heading is that they may concern 'one-off' ideas, and in such cases, the development of generalized techniques may be considered rather absurd, even if possible. Some of the objectives within the fourth type can be converted to the above-mentioned types of performances. Take the example of a design task concerning a school building that will be used for the education of disabled people. The evaluations that check for the special physical configurations of such a building may be carried out through parametric definitions with regard to building standards and codes; therefore, can be considered under the first heading. However, an architect may imbue such a task with an additional mission to communicate a

message about the social status of the disabled people. The symbolic mission that is added by the architect is peculiar to the situation and there is no predefined standard for carrying it out.

The diagram in Figure 2.16 brings together and summarizes the basic issues that have been discussed until this point. The column on the left side summarizes the main problems that human or artificial design agents have to face within design situations. A summary of the strategies and capabilities that human designers utilize for tackling these problems are given on the right side. Design situations are both complex and vague. Human designers employ both everyday intelligence and expert skills to dynamically (re)frame the situations, strategically focus on subsets of tasks, and propose tentative solutions, throughout a co-evolution of problems and solutions. The skills summarized on the right side are the very skills that artificial design agents would require to carry out design.

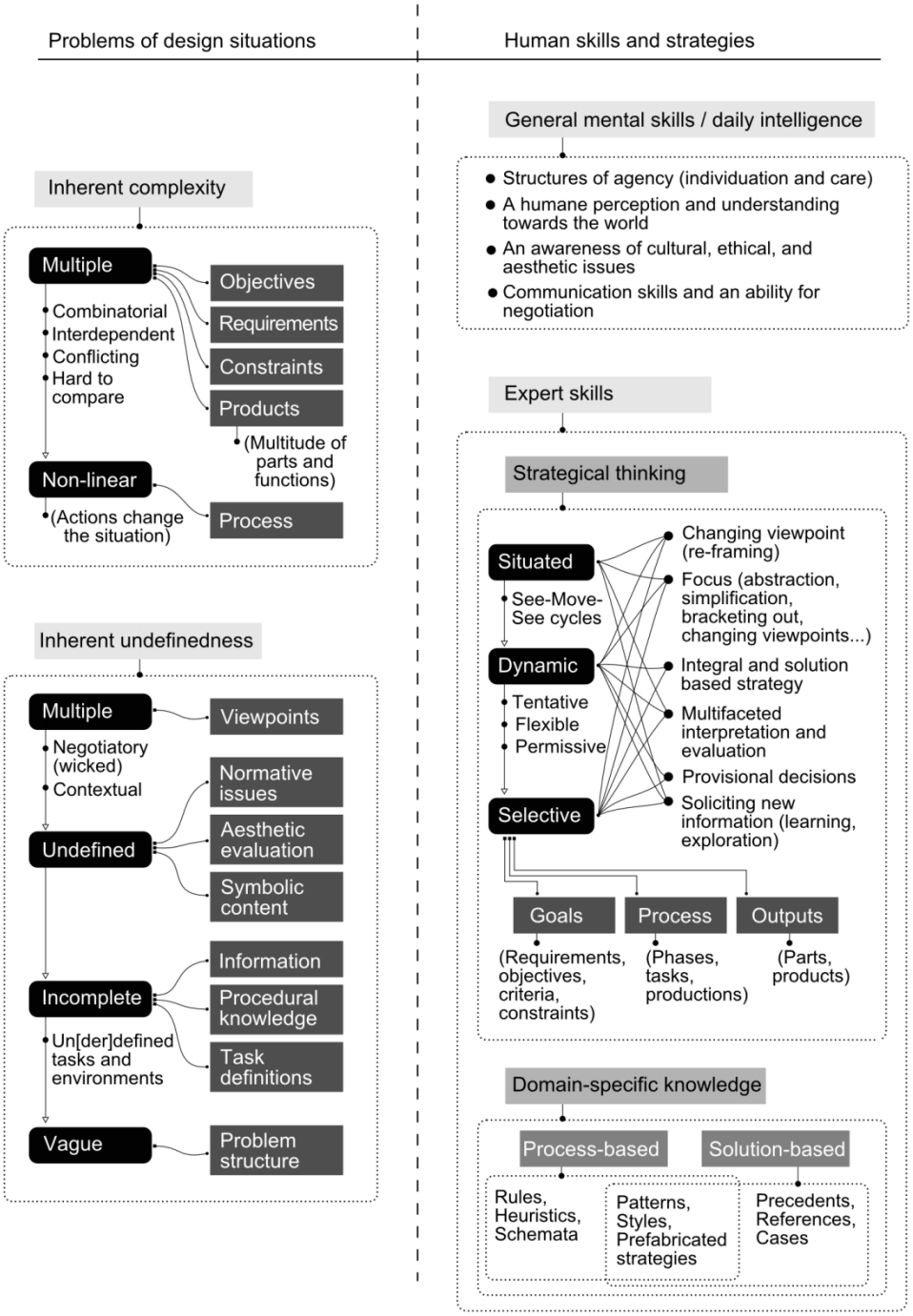


Figure 2.16 : A summary of the basic problems of design situations (left side) and the strategies and skills of human designers (right side).

2.5 The Research Program of Artificial / Autonomous / Automated Design

In an illustrative passage, Geoffrey Broadbent presents an early interpretation for the motivations behind the studies on Artificial / Autonomous / Automated Design, or shortly, AD:

> More and more over the past two hundred years, the various functions of building design have been separated out . . . the architect's task gradually shrank; he was left with the 'soft-edged' aspects of building design, those things which could not be quantified but relied instead on his skill and judgment as an aesthete and a gentleman. He was concerned, above all, with matters of form, proportion, colour and texture—the raw material, as it were, of visual delight. . . . Increasingly, however, it seemed probable that these too could be quantified. Colour, for instance, could be specified in terms of hue, value and chroma; experiments could be set up in which psychological response to colour could be quantified, and thus a further aspect of the architect's remaining task would be eroded. Form, proportion, texture and so on could follow so that the architect might be left, literally, with nothing to do (Broadbent, 1973, p. xi).

The underlying idea here is that when quantified in a measurable form, or defined in the form of a series of explicit interconnected rules or symbolic expressions, those remaining aspects of the design processes that are currently reserved to the human designers could be gradually taken over by autonomous agents. In Broadbent's expression, AD is comprehended with (or perhaps accused for) an emphasis on the explicit, symbolic, and measurable kinds of knowledge and reasoning.

From this point of view, Broadbendt (1973, p. xiii) points to the idea that the proponents of AD could offer powerful tools for decision-making only when one has data in quantifiable form. However, for many kinds of design, it is important not just to be technically competent but also to have a well-developed aesthetic appreciation, which is still considered an elusive notion. Because the cultural issues that the designers have to deal with do not seem to be amenable to a simple method of quantification, Broadbendt thought it was clear that the design of architecture could never be a matter of completely automated decision-making.

Yet, in light of the criticisms against early AI studies, therefore, also against early AD, it appears that rather than rendering the soft aspects of design explicit or definite, the basic problem of AD is to utilize two types of reasoning together. On the one hand, an ability for sub-symbolic recognition, knowing, and learning within worldly situations; and on the other hand, the rather explicit and symbolic aspects of

reasoning that may involve language, rules, calculations, logic, and the like. Therefore, a fuller expression of AD's quest would involve unexplicit types of recognition and learning methods (e.g. artificial neural networks, support vector machines, etc.), datamining techniques, soft computing approaches (e.g. evolutionary algorithms), and situated, online agents.

Either only with explicit symbolic definitions and measurements, or with the help of sub-symbolic, unexplicit, and situated technologies, or with combinations of both, the task of AD appears as a ground-up, constructive enterprise; gradually taking over the burden of the human designers and ultimately leading to intelligent artificial design agents that would carry out a set of design tasks autonomously. This quest for completely autonomous design agents may be called strong-AD, with reference to the duality of strong-AI vs. weak-AI.

Two main senses have been attributed to the term strong-AI. Firstly, strong-AI has been associated with the claim that an appropriately programmed computer can literally be a mind and can think in the sense that humans think. The second and weaker claim associated with the term simply states that computers can carry out all major mental tasks at least as well as humans do; but without the requirement that they do this in the exact sense as humans do. In either case, strong-AI requires the development of general intelligence or global and general-purpose thought processes to cope with the highly contextual human endeavors.

On the other hand, weak-AI has come to be associated with attempts to build programs that aid humans, rather than duplicating their mental activities. Regarding its aims, weak-AI has shown practical achievements, whereas the search for strong-AI, if ever possible, will have to continue for a rather long time (Nilsson, 2009, p. 380, 381).

The parallel distinction between fully automated design (strong-AD) and computational design aid (weak-AD) might help clarify several points in our discussion. While the enthusiasm for weaker claims for AD has never waned, a suspicion towards strong-AD—albeit over mostly general and sometimes superficial terms—has always been widespread. This suspicion has appeared almost concurrently with the enthusiasm for AD's meager accomplishments.

One basic criticism towards strong-AD concerns the ill-defined character of design. Not only design problems, but indeed most of the real world problems are ill-defined and they pose serious problems for practical AI applications (Simon, 1973; 1996). Consequently, the earliest AI studies mainly tackled well-defined toy problems. If AI were limited to these techniques, design problems would have to be left outside. Another basic criticism concerns one of the sources of this ill-defined character, i.e., the need to embrace so many different kinds of thinking and knowledge:

> Scientists may be able to do their job perfectly well without even the faintest notion of how artists think, and artists for their part certainly do not depend upon scientific method. For designers life is not so simple, they must appreciate the nature of both art and science and in addition they must be able to design! (Lawson, 2005, pp. 13, 14).

Taking the above idea as another reason why it seems unlikely that computers can ever be enabled to design in the sense that humans do, Lawson and Dorst (2009) claim that there is something "essentially human" in design. They also add that, the fact that design remains a human activity beyond the capability of AI poses interesting challenges to the computational theory of mind. This last point underlines the importance of a field of AD, beyond practical aims.

There are also representational issues in both knowledge representation and task and state-space definition that pose specific problems for design fields. Designers apply tacit and context-specific knowledge and rules simultaneously with forms of visual thinking. They think by manipulating graphical information, which is difficult to encode in conventional symbolic systems as employed by most current AI (Lawson and Dorst, 2009, p. 104).

These basic remarks are sufficient to indicate that design situations are complicated, and interpretation and action within these situations are highly context dependent (Gedenryd, 1998; Schön, 1983). A mapping of the basic components of this complexity has been given in Figure 2.15. In a basic design situation, there are agents (or actors), problems, and solutions (or outputs), which transform together within a complex and interrelated evolution. Agents produce and re-produce a multitude of viewpoints (or framings) while outputs are co-evolving with the problems that are controlled by a large amount of possibly conflicting objectives and constraints.

The design context contains almost every aspect of the human world—cultural, physical, psychological, and so on—and requires both a general everyday

intelligence and an expertise on a particular design profession. The determined, undetermined, vague, unique, value-laden, conflicted, simple, and complex problems are all parts of design situations. Specifically in architecture, in any design process, each of these characteristics is observed while the process unfolds from its first conceptions to the construction phases. The types of design tasks, which have to be tackled for a convincing design solution to appear include the conceptual problem formation stages, where the designers need extensive understanding of every aspect of culture as well as a complete ability to understand and manipulate shapes and objects. A creative and sensitive design proposal, which is in dialogue with its cultural, political, economic, and historical context, does not seem to be attainable with a mere reuse of existing solutions or with a context-free application of predefined rules and constraints.

Problem reduction through rigorous formal representation is not a valid approach towards AD, which has been shown by the practical and interpretative failure of systematic design methods and problem-solving approaches to design. Design requires constant reappreciation of the situation and a redefinition of the problem, or task environment at each step. Thus, design agents should function in an online manner. While automation of the problem formulation step is the weakest side of current AI approaches in design, one of the most important abilities in design is this (re)appreciation, (re)formulation, or (re)framing action.

In a real design process, goals, representations, and transformation operators are often only tentatively chosen or determined, they are not fixed, and they are up to constant change. Indeed, design is the dynamic process of devising these definitions in an ever-changing task environment, so that for each design process unique chains of operators, goal complexes, and representations are interweaved following rather tactical, ad hoc moves. Compare this idea with an alternative conception where the tools, representations, and goals are being chosen from a rather stable repository. This second conception appears easy to implement, however the crucial problem is determining when and how some design element will be used. Context-free prefabricated strategies could be useful, but they are not sufficient. Because without knowing which strategy to apply and how to adapt a generic strategy to a specific context, the context-free definition of a strategy will be useless, and it is out of the question to try to define a repository of all strategies that could be needed.

Design intelligence requires coping with incomplete knowledge or even with an inherent vagueness within daily situations. Because designers navigate within a human world, design intelligence requires an ability to appreciate almost every aspect of human culture. Having a general understanding of daily issues, discerning contextual differentiations, developing an arsenal of expert strategies, and possessing motivations to deal with human situations are prerequisites for strong-AD. Therefore, a level of machine intelligence almost reaching the human mind has to be attained before entrusting design activity altogether to the machines. Consequently, design appears as a challenge for AI in general, and a quest for strong-AD would require no less than the accomplishment of the claims of strong-AI. In this sense, it will be claimed that, Strong-AD is AI-complete: the problem is within the class of the hardest problems of AI.

This means that the developments in AD will be strongly linked with the developments in intelligent agents in general, and especially with the development of technologies that are directed towards a rather general daily intelligence that is capable of dealing with cultural and psychological issues at least on a basic level. However, the development of intelligent agents in ways that are significant for Computational Design systems has encountered difficult problems as will be discussed below with reference to the criticisms of Hubert Dreyfus.

In his influential criticisms towards early AI studies, Dreyfus (1972) claimed that these research efforts frequently displayed initial apparent success on restricted toy problems and artificial worlds, to be followed by a failure in scaling the systems towards real world problems. As an explanation for this failure pattern, he further claimed that these approaches had been based on four philosophical assumptions, all of which were fallacious. Following the "biological assumption", AI researchers supposed that the brain was processing information in discrete operations by way of some biological equivalent of on/off switches. According to the "psychological assumption", the mind could be viewed as a device operating on bits of information according to formal rules. The "epistemological assumption" amounted to the supposition that all knowledge could be expressed within a formalized form. Finally, the assumption that the world could be exhaustively analyzed in terms of determinate data or atomic facts was called the "ontological assumption".

In order to construct an alternative picture of human mental life, Dreyfus appealed to his readings on Heidegger, Wittgenstein, and Merleau-Ponty. According to Dreyfus' alternative picture, any system, which would equal human performance, had to be able to "1. Distinguish the essential from the inessential features of a particular instance of a pattern, 2. Use cues which remain on the fringes of consciousness, 3. Take account of the context, 4. Perceive the individual as typical, i.e., situate the individual with respect to a paradigm case" (Dreyfus, 1972, p.40).

For Dreyfus (1972), a supposedly nonprogrammable ability was the ability to deal with ambiguous situations and concerned context-dependent activities like language translation. The ability to narrow down the spectrum of possible meanings as much as the situation requires was called "ambiguity tolerance" by Dreyfus, and was set in opposition to the pursuit of context-free precision or formalization. The latter, rather than coping with ambiguity, was trying to eliminate it altogether (Dreyfus, 1972, p. 19). Ambiguity tolerance, in turn, was revealing another fundamental form of human information processing, namely, "fringe consciousness", which took account of cues in the context without explicit consideration (Dreyfus, 1972). At any time during a mental operation, an indefinite group of elements resided on the fringes of consciousness that were ready to be selected whenever they would be relevant for the situation. Fringe consciousness appeared to be working together with an ability to discriminate essential and inessential constituents of a situation. Dreyfus contrasted this ability, together with fringe consciousness, with the trial-error based search mechanisms (Dreyfus, 1972, p.24).

Examining a protocol study of a chess player, which was previously cited by Simon and Newell, Dreyfus (1972) went on to develop an alternative interpretation of the protocol and claimed that "all protocols testify that chess involves two kinds of behavior: (1) zeroing in, by means of the overall organization of the perceptual field, on an area formerly on the fringes of consciousness, and which other areas still on the fringes of consciousness make interesting; and (2) counting out explicit alternatives" (Dreyfus, 1972, p.18). The novelty in this formulation was the dependency of the explicit reasoning process on an underlying global recognition faculty. Parallel to this conception, expertise consisted of an interrelation of two types of ability, namely, "knowing-how" and "knowing-that". Knowing-that was the explicit, step-by-step, conscious type of expert knowledge, while knowing-how

corresponded to the tacit aspect of daily skills (Dreyfus, Dreyfus, and Athanasiou, 1986).

Finally, considering pattern recognition tasks, Dreyfus concluded that recognition of even simple patterns was requiring recourse to a special combination of insight, fringe consciousness, and ambiguity tolerance. He called this combined ability "perspicuous grouping" and contrasted it with the attempts to describe a pattern in terms of a list of specific traits (Dreyfus, 1972, p.32).

In his general conclusion to his examination, he invited the AI researchers to stop what they had been doing and start to look for the deeper structure of the problem:

> If the alchemist had stopped poring over his retorts and pentagrams and had spent his time looking for the deeper structure of the problem, as primitive man took his eyes off the moon, came out of the trees, and discovered fire and the wheel, things would have been set moving in a more promising direction. After all, three hundred years after the alchemists we did get gold from lead (and we have landed on the moon), but only after we abandoned work on the alchemic level, and worked to understand the chemical level and the even deeper nuclear level instead (Dreyfus, 1972, p.217).

Indeed, a group of AI researchers adopted new research paths in line with all major criticisms of Dreyfus. These new research paths comprised embodied and interactive (situated) cognition, connectionist approaches, probabilistic methods, machine learning, and the approaches grouped under the name 'soft computing'.

Throughout this transformation of the field, Dreyfus revised and extended his criticisms with several subsequent editions of his 1972 book. In his introduction to the 1999 edition, he came to distinguish between early AI and novel approaches. He considered expert systems—against which he had published another influential volume together with his brother Stuart Dreyfus (Dreyfus, Dreyfus, and Athanasiou, 1986)—an offshoot of the early AI, and assumed a more positive tone towards connectionist, embodied, and sub-symbolic approaches. However, eventually, he claimed that even with these new approaches, systems that understood and behaved like humans were still lacking: "It looks likely that the neglected and then revived connectionist approach is merely getting its deserved chance to fail" (Dreyfus, 1999, p.xxxviii).

Following the criticisms of Dreyfus, it can be claimed that, if there has been a heritage of technical rationality, which has been manifested by the early AI studies,

there has also always been an undercurrent complementing the picture, which constantly brings in the aspects that have been omitted by the first. As mentioned above, Dreyfus' criticisms were not predominantly based on scientific research, but rather on a philosophical perspective. These remarks set us in line with a sustained and holistic criticism against a reductive view of human mental life. We can expand Dreyfus' explicit references for this alternative lineage, i.e., Heidegger, later Wittgenstein, and Merleau-Ponty, to include existential philosophies, and phenomenological and hermeneutical traditions in general.

This thesis regards this alternative conception as complementary to the technical rationality, without recourse to a mystical sensibility that is also widespread. The insightfulness of such criticism can be measured by its obvious practical influence on the transformation of AI in general. With the advent of new AI technologies, which will be briefly introduced in the following pages, the two apparently incompatible views may now be reconciled within the same picture of the mind, as symbolic and sub-symbolic aspects of thinking. These two aspects have to cooperate for design thinking to occur.

This picture explains the practical failure of the one-sided approach towards formalization of basic procedures of design into well-defined task specifications or mathematical formalisms. It also rules out technologies like shape grammars that depend solely on strict formal definitions and transformation rules, as well as expert systems, which depend on rule-like explicit knowledge; at least when these are used without recourse to additional technologies.

This is also the reason why Donald Schön's notion of reflective practice explicitly opposes the dominance of a technical rationality. Accordingly, we can claim that this is the overall reason why design is not an extension of problem-solving. Design may partially involve explicit reasoning, measured definitions, rules and heuristics, and even strictly logical thinking, but it also involves types of cognition that are fundamentally different from these, and it is this second type of thinking, which predominantly determines design cognition. This is also the main reason why there is no textbook for design education and there are no overarching theories or rules or context-free heuristics that designers rely upon in practice.

However, the utility of the problem-solving approach has to be better outlined for a more balanced assessment. Even when we set aside reductive philosophical assumptions, a reductive path of research may stand out simply because of practical reasons. If there are repeating problem types in a field, and if reliable solutions are within reach for each of these problem types, then whenever a new problem is identified as falling under one of these already defined types of problems, then existing techniques can be used to solve the new problem. This would remove the need to develop unique methods for each new problem, and would enable a focusing of research efforts towards a few problem types. Such a well-studied solution procedure to a problem type appears as a good example of what can be called a prefabricated solution strategy[13].

In engineering design, and in the engineering oriented aspects of architectural design, this approach is readily applicable. However, for this thesis, it is more interesting to observe that even within ill-defined situations of design, human designers often appear to be applying kinds of prefabricated strategies, whenever they detect recurrent situations and problems. Use of precedents, patterns, and style has been discussed above as examples of this occurrence. When reconsidered in light of Dreyfus' criticisms, there are two crucial requirements for a prefabricated strategy to be applied within an architectural design process. The first is an ability to recognize the situation where a particular strategy would fit. To this basic situational recognition ability, another ability should be added, that is, an ability to sensitively adapt this prefabricated strategy within a new situation. Case-based design set forth to tackle this issue and failed to deliver a practical solution. We can claim that its failure was not based on its overall aim, but on the underlying assumptions about intelligence, which were very much in line with the early AI paradigm.

As we have stated above, at least within complex design fields like architecture, strong-AD, i.e., replacing the human designer altogether, does not seem to be on the agenda of researchers today. However, a clarification of the requirements for such research might be beneficial for subsequent studies.

---

[13] For such problem types and proposed solution procedures in engineering fields, see Michalewicz and Fogel, 2004.

AD's basic assumption should be that, current human strategies and behaviors in design are the proper responses to a design situation. AD's second assumption states that, design computation takes place in a design environment where the humans and CAD tools operate together. These constitute varied forms of human-tool complexes. Assuming a weak position towards AD, Computational Design studies should tackle these problems and offer solutions within human-tool complexes. In these human-tool complexes, the role of the CAD tools is usually taken as aids for the human designers, who act as the principle manager of the process.

While weak-AD is the practical paradigm for Computational Design, strong-AD may remain an ultimate target, or the limit to converge. The only way towards such an ultimate aim is the cooperative human-tool complexes that would work together, through which the artificial partner would be learning from the human agent. However, there is an apparent dilemma against this mutualist learning approach, which again has been brought forward by Dreyfus:

> One needs a learning device that shares enough human concerns and human structure to learn to generalize the way human beings do. And as improbable as it was that one could build a device that could capture our humanity in a physical symbol system, it seems at least as unlikely that one could build a device sufficiently like us to act and learn in our world (Dreyfus, 1999, pp. xlv-xlvi).

There are two requirements for a solution to this problem. The first is the development and incorporation of the machine-learning technologies. The second requirement is the voluntary acceptance of an artificial agent within this cooperation, as a partner by the human agent.

With regard to the new AI technologies, as mentioned above, partly because of Dreyfus' criticisms, a group of AI and cognitive science researchers started to defend situated and embodied approaches for studying and developing intelligent systems (Brooks, 1999). A set of claims has been guiding this line of research, although each researcher may be adopting her own combination of these claims (Wilson, 2002). Within this paradigm, cognition is claimed to be situated, action-oriented, body-based, and time pressured. Moreover, it is assumed to be extended to the agent's environment, to the extent that the environment becomes part of the cognitive system (Wilson, 2002). This type of paradigm shift is crucial for Computational Design, at least when it concerns architecture, because human-like spatial perception is a

requirement for understanding most architectural concepts and for sensitive spatial evaluation.

Another important development for Computational Design is novel machine-learning techniques. As Dreyfus' criticisms revealed, symbolic approaches have failed in developing learning systems. Therefore, new approaches that do not resort to explicit knowledge were being required. Artificial neural networks (ANN), which were loosely been modeled on how real brains work, emerged as a potential solution to this problem. ANNs embody classificatory knowledge implicitly in their variables (Bishop, 2006). While it is not easy to visualize or understand what is going on in their recognition processes in an intuitive manner, or through a short list of natural language expressions, they are robust and flexible recognizers, and their working resembles human recognition, at least far better than symbolic systems. These types of systems can be seen as sub-symbolic systems, underlying and supporting symbolic (language-based and mathematical thinking), conscious, and deliberate mental processes, in the sense that symbolic and sub-symbolic aspects support each other in carrying out mental tasks.

ANNs are artificial learning systems and there are supervised and unsupervised types of learning (Bishop, 2006). Supervised learning approach can roughly be seen as a process where the questions and answers are simultaneously provided to a system. In the unsupervised approaches, the learning system is expected to find the relevant information by itself from a set of examples or by datamining through the data. Combinations of these two approaches are also possible. Another option is to use online agents that learn while they operate. This brings us to the second requirement, the acceptance of an artificial agent within design processes. We will discuss this with reference to Nicholas Negroponte's research path and views.

Starting in mid 1960's, with his "Architecture Machine Group" (AMG), Nicholas Negroponte laid down a pioneering research programme, which has identified some basic requirements of strong-AD. The AMG had set its aims as demonstrating or finding the paths for machine intelligence in the context of architectural design (Negroponte, 1970). The central ideal of AMG was an architecture machine, which could understand human metaphors, solicit information on its own, acquire experiences, talk to a wide variety of people, and improve over time (Negroponte 1970; Cross, 1977). Towards this end, the group has built a number of small

machines and devices. For example, in their SEE project, which dealt with machine vision, a computer agent, while looking at three-dimensional arrangements of blocks by means of a video lens, was trying to draw what it was seeing. The GROPE project involved a mobile light sensor to investigate printed maps for interesting features. GREET was a doorway which recognized whoever passed through. HUNCH project aimed at programming the computer to recognize sketches, and SEEK was a model environment consisting of small cubes that could be arranged by a robot arm (Cross, 1977).

Although their aim was to develop design assistants, rather than autonomous design agents, Negroponte foresaw—in line with the requirements that were brought forward by Dreyfus' above-mentioned dilemma—that the architecture machine could be nothing less than an intuitive and insightful companion for the human designer. The reason for this rather ambitious projection was that, to become a companion to the human designer, the autonomous agent had to understand her within real situations, where any procedures or set of rules would be contextual. A mechanism had to recognize and understand its context before carrying out an operation. Moreover, a machine had to be able to discern changes in meaning brought about by changes in the context, and to do this, it had to have a sophisticated set of sensors, effectors, and processors to view the real world directly and indirectly (Negroponte, 1970). This explains the wide scope of the AMG project, which had to encompass the whole of the field of AI:

> . . . we, the architecture machine group at M.I.T., are embarking on the construction of a machine that can work with missing information. To do this, an architecture machine must understand our metaphors, must solicit information on its own, must acquire experiences, must talk to a wide variety of people, must improve over time, and must be intelligent. It must recognize context, particularly changes in goals and changes in meaning brought about by changes in context (Negroponte, 1970, pp. 119-121).

It should also be noted that, according to Negroponte, the artificial agent and the human had to evolve together, to get to know each other to a level of mutual understanding. This means, the frontier has remained at the very spot where Computational Design has started. The overall task is to develop more intelligent assistants that would be able to work smoothly with the human agents while learning from them; in other words, to develop human-tool complexes that would function

together; better or at least more pleasurably, so that someday, through a gradual development, autonomous artificial design agents may be possible.

2.6 design_proxy: An Integrated Approach for Draft Making Design Assistants

Following the considerations of the previous sections, a viable practical target for AD appears as to develop artificial assistants that can generate draft proposals for design tasks. The aim is stated as generating drafts, instead of already developed design products, because of the limitations of current AI techniques. Although techniques are available for automating a series of well-defined tasks within design, such as optimization methods for the later phases of design processes, these are not suitable for un(der)defined tasks. In contrast to problem-solving, design requires both an everyday intelligence and expert skills towards design fields (see Figure 2.16), whereas computational technologies for both are still on a primitive level. Nevertheless, generating draft proposals—which is already a difficult task—appears achievable with the help of existing techniques such as Evolutionary Computation (EC) and machine-learning methods.

With a long-term perspective for AD studies, a draft making computational design assistant should be, first and foremost, readily usable; not simply experimental, but usable within real situations, so that it may start to be developed. Draft making assistants may attain their places within regular design processes as part of a mechanism of brainstorming. Through the incorporation of such assistants, instead of starting each new task from scratch, human designers could be able to entrust a further development of their initial ideas to their computational assistants, while they are spending their time on other tasks, or simply resting. When the drafts are ready on their desks, they could move forward by critically examining and revising the draft proposals. This would mean both analyzing a situation through examples and moving forward through proposals; therefore, draft making assistants could function as both analysis and exploration tools.

The previous chapters put forward a series of requirements for such assistants. The "design_proxy" (d_p) approach is proposed as an integrated response to these requirements. It consists of a set of complementary tenets, or guidelines, which are (Figure 2.17):

1. Flexible and relaxed task definitions and representations (instead of strict formalisms).
2. Intuitive interfaces that make use of usual design media.
3. Evaluation of solution proposals through their similarity to given examples.
4. A dynamic evolutionary approach for solution generation.

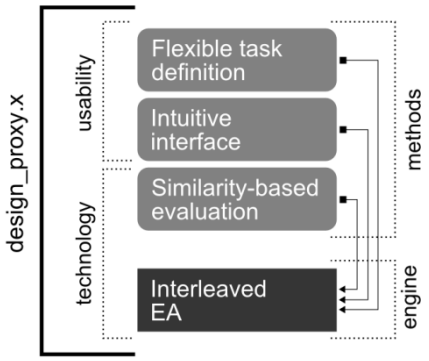


Figure 2.17 : Basic constituents of the design_proxy approach.

Although it might at first appear as a syncretic collection of gratuitous elements, the main tenets of the design_proxy approach constitute a set of integrated and complementary techniques, developed both with regard to the considerations of the previous sections and as a result of practical experimentations. In addition to the brief descriptions and reasons that will be given below, the applications of the approach that will be presented in Chapter 4 will enable a more in-depth exposition of the functioning of these tenets, and a more detailed discussion of the rationale behind these.

The demand for relaxed task definitions, and the ensuing flexible and permissive problem representations are the direct consequences of determining the practical target as draft-making. Moreover, such relaxed character is expected to contribute in the applicability, adaptability, and versatility, hence in the robustness and reusability of an artificial assistant.

An intuitive input-output interface through usual design media is suggested to enable an artificial design assistant to be easily accepted within a regular design process of human teams. A design assistant should not assert its own workflow, but should be able to be simply added into existing design workflows. Design assistants that would communicate and operate through usual design media, instead of bringing forward

their own dedicated interfaces, would be easier to learn, to use, and to adapt to. In our application cases, this medium is mainly graphics, for both input and output, and mostly within a sketching functionality.

Evaluation through similarity is the core tenet of the design_proxy approach. It amounts to the evaluation of a design proposal with respect to its similarity to a given example, or a paradigm case, in terms of a specific feature. It is based on the idea that the information embedded within existing examples could be used to answer questions regarding design rationale or the evaluation of proposals. As such, it is a robust method for comparing alternatives, and it brings forward possibilities for information gathering, analysis, use, and also for learning systems. A similarity-based evaluation method is open for learning on two levels: first, its database can be constantly extended; secondly, new learning and analysis methods can be gradually added, to improve both its usage of existing databases and its learning capabilities.

Using the information that resides in existing examples is highly congruent with the two previous tenets. When existing examples are used through similarity measures, this enables the development of an example-based interface that simply uses the traditional representations of these examples; through which, preferences can be defined by simply approving or rejecting precedents. The products of the system can also use the same representation enabling a consistent manner of communication. This also dispenses with the need for explicit definition of fixed objective functions and opens the evaluation process for dynamism through a selection of new examples or a selective focus on relevant information. The specific features and methods for similarity measurement have to be determined before a process starts, yet each new example or group of examples defines a new objective in a dynamic manner. Indeed, many questions that appear throughout a design process may be answered with recourse to collections of existing examples, so that instead of an initial problem-setting effort, a dynamic method may be developed through constant recourse to sets of examples. These collections of examples can be gathered together through collaborative information gathering and sharing.

The applications of design_proxy develop and demonstrate aspects of the similarity-based evaluation, presenting its capabilities for different tasks. Through the presentation of these applications both the details of the approach and theoretical issues will be discussed in a more concrete setting.

The final requirement of the design_proxy approach is a potentially parallel and hierarchical generative mechanism that would bring together the above-mentioned methods within a dynamic and adaptive development process. The Interleaved Evolutionary Algorithm (Interleaved EA) is a novel EA, specifically developed as the engine of a design_proxy-based assistant. It depends on the successive operation of single-objective evolutionary processes over the same population with regard to the feedback gained from the progression of the evolutionary process. As such, it enables the use of separate operators and settings for different objectives, and is a dynamic, complex, adaptive, and multi-objective solution generation process (It will be defined in more detail at the end of Chapter 3, after the main concepts of Evolutionary Computation are introduced).

The design_proxy approach is envisioned as adaptable to various design tasks. In this study, two applications of design_proxy approach will be presented. Each of the applications is used for the development and testing of different aspects of the complicated integration demanded by the design_proxy approach. The "d_p.graphics" application generates graphic patterns, and is rather used for demonstrating and testing the technical aspects of the design_proxy with regard to its integration with the Interleaved EA. On the other hand, the d_p.layout application undertakes to develop a concrete and usable architectural layout design assistant through the design_proxy approach. This application is used to tackle a real world problem, i.e., layout design, in order to investigate, develop, and demonstrate the usability demands.

Current applications of the design_proxy approach operate at the level of a single task. However, complex fields such as architecture demand more complicated assistants that should carry forward a set of tasks in parallel, so that an overall design task may be tackled. This can be possible with the use of hierarchical and parallel evolutionary settings. These potentials will be first discussed in terms of Evolutionary Computation and the Interleaved EA in Chapter 3, where a scheme will be presented for higher-level design_proxy assistants. Furthermore, in Chapter 4, the "Architectural Stem Cells" (ASC) framework will be presented for illustrating how a higher-level design assistant for specifically architecture could be developed. The ASC will situate the d_p.layout within architecture, while bringing forward a detailed

research agenda that targets the development of a comprehensive, multi-layered, dynamic architectural design assistant.

2.7 Conclusion

This chapter was dedicated to a detailed theoretical outline concerning the relationship of design processes and Computational Design. With an aim to reveal, support, and discuss the aims and directions adopted in the thesis, as well as to set forth a plausible research program for prospective studies, this theoretical investigation tried to find out why previous approaches have had problems in satisfying the quest for artificial design intelligence.

The chapter proposed a non-reductive mapping of the basic constituents of a design situation (Figure 2.15). The mapping involves agents who access the co-evolving problem and solution spaces through viewpoints, interpretations, framings, and actions. Rather than constituting a list of basic atomic actions, each of these terms designates a separate aspect of a complex interaction.

In order to complement the technically oriented approaches to performance, a fourfold classification was proposed for the multifarious evaluations required within design situations. The basic categories of performances comprise prudence, well-defined performance, underdefined performance, and undefined performance types (Section 2.4.10).

As an important output of the theoretical investigation, the basic difficulties encountered in design situations and the corresponding strategies and techniques employed by human designers were summarized with a mapping (Figure 2.16). The first group of problems concerns the "inherent complexity" of design situations in terms of a multiplicity of objectives, requirements, constraints, parts, and productions. A second group of difficulties arises in the form of an "inherent undefinedness" because of the multiplicity of viewpoints and tastes, which result in un(der)defined tasks, incomplete information, and the contextual character of design knowledge. Design situations are essentially undefined, because until a design process starts to unfold, the required negotiations cannot take place, the tasks cannot be defined, and relevant information cannot be determined. This complex and

undefined nature of design situations is the basis of our position against reductive strategies.

The second part of this mapping (Figure 2.16) attempts to identify the strategies assumed by human designers encountering these challenges. The basis of these strategies is the dynamic approaches built over a substrate of daily, embodied, situated intelligence. Interpretation and domain-specific expertise is to be built over this daily substrate and will involve, on the one hand, prefabricated, reusable strategies such as heuristics, schemata, patterns, precedents, style, etc., and on the other hand, dynamic strategies such as tentative proposals and viewpoints (re-framing); changeable motivations, goals, objectives, criteria, and constraints; continuous learning (eliciting new information where required); and dynamic structuring of the process through a dynamic focus (decomposition + integration).

The identified human strategies also reveal the required capabilities for artificial design agents. It appears that to carry out the integral and dynamic exploration characteristic of design, a design agent requires building design expertise over an interpretative and evaluative daily understanding. The attempt at an identification of the difficulties of design and the solutions for these difficulties provides a basis for a critical examination of the previous studies as well as for the development of a research program for Artificial / Autonomous / Automated Design (AD).

At this point, the chapter focused on a research program, whose ultimate aim would be total or partial, i.e., strong or weak AD. The general component of this response is a research program for AD, for which a set of basic tenets has been proposed: (1) Current human strategies and behaviors in design are the proper response to design situations. (2) Design computation takes place in a design environment where the humans and CAD tools operate together. (3) Because strong-AD is amongst the hardest AI problems, weak-AD is the practical paradigm for AD studies; however, strong-AD may remain an ultimate target.

The overall task that appears after these examinations is to develop intelligent artificial assistants that would be able to operate together with the human agents while learning from them. This is also valid for EC based systems. The importance of this plain proposal is the displacement of the emphasis from rigor or technical

success to usability, which would require ease-of-use and pleasure no less than utility, scientific rigor, or technical competence.

Finally, the practical response for these requirements was an integrated approach for guiding the development of artificial design assistants[14], i.e., the "design_proxy" approach. It consists of a set of complementary tenets, or guidelines, which are (see Figure 2.17): (1) Flexible and relaxed task definitions and representations (instead of strict formalisms). (2) Intuitive interfaces that make use of usual design media. (3) Evaluation of solution proposals through their similarity to given examples. (4) A dynamic evolutionary approach for solution generation.

[14] An example of such a research approach is proposed by Pfeifer and Bongard (2006). Coming from a practical background, they propose general guidelines for robotics and AI research. Although Pfeifer and Bongard's approach has not been a direct precursor or inspiration for it, the design_proxy approach stemmed from a similar practical need for overall guidelines.

3. EVOLUTIONARY COMPUTATION FOR DESIGN

This chapter concerns theoretical and technical issues regarding Evolutionary Computation (EC) and introduces the "Interleaved Evolutionary Algorithm" (Interleaved EA, IEA), which is a complex, dynamic, adaptive, and multi-objective evolutionary algorithm devised for design.

The chapter will start with an overall introduction to the basics of EC, after which a brief history of EC will be given. On a more specific level, EC usage in art, design, and architecture will be exemplified, and basic issues, requirements, advantages, and drawbacks of EC will be discussed. As part of the conceptual background for the Interleaved EA, several schemes, illustrating complex and hierarchical EAs, will be presented and discussed. The chapter ends with a description of the Interleaved EA.

3.1 What is Evolutionary Computation?

Evolutionary Computation (EC) denotes a family of techniques within computer science, which is inspired by the processes of biological evolution. EC typically exploits mechanisms like variation, reproduction, and selection and is often used for problem-solving and optimization, especially when the problems do not lend themselves to easily applicable algorithmic procedures.

There are many different variants of evolutionary algorithms (EAs), which are united by a common underlying idea: “given a population of individuals, the environmental pressure causes natural selection (survival of the fittest), which causes a rise in the fitness of the population” (Eiben and Smith, 2003, p. 15). In any population of organisms, the individual organisms that constitute that population exhibit similarities as well as variations. They are of the same type, same family, in the sense that they display different traits while they remain comparable. When there is a context that renders a specific trait more advantageous for survival, the mechanisms that are responsible for this trait have a higher possibility for being transferred to future generations. Thus, evolution pursues these minor differences, which increase the ability for survival of the individuals. While individuals survive for a limited

amount of time, the mechanisms that are responsible for the successful traits may subsist for a longer time by being embedded within individuals. These underlying generative mechanisms are generally called the genes.

In a simple case, an artificial evolutionary process starts with a population of individuals, which are seen as the solution candidates for a problem. These candidates are initiated either randomly or purposefully, using domain-specific knowledge. Each round of operations is called a generation, following the biological metaphor. The individuals of a generation are evaluated using an evaluation mechanism, which can take the form of interactive qualitative evaluation, a combination of fitness functions, or satisfaction of a set of constraints. Evaluation yields fitness measures for each individual. Using these fitness values, the individuals can be compared with each other.

This comparison constitutes the basis of the selection mechanisms during the process. Selection mechanisms act as a force that pushes for quality (Eiben and Smith, 2003, p. 16). One selection threshold concerns the selection of parents that will reproduce. Choosing better parents possibly results in better offspring. A second threshold for selection involves choosing which of the individuals will 'die' and which will 'live' through the next generation. Usually, a smaller set of individuals is selected amongst the candidates for the next generation. This way the least fit individuals are terminated and more promising individuals are selected. Sometimes individuals may be terminated just because they exceed a certain age threshold.

After the parents are selected, new individuals are created either by mutation of one individual or recombination of two or more parents. Mutation and recombination act as variation operators; they create the necessary diversity and facilitate novelty within an evolutionary process (Eiben and Smith, 2003, p. 16).

Each individual has a 'genotype', reminiscent of biological DNA, and the mutation and recombination operations are generally carried out over this information. The fitness evaluation on the other hand, is mostly carried out over the 'phenotype', which denotes the fully-grown individual, built according to the information presented in the genotype. The mechanism for mapping the genotype to the phenotype may be external to the main evolutionary process.

The evolutionary process is iterated until a sufficient solution is attained, a specific condition is met, or a computational limit is reached. The process can be terminated when the improvement of fitness values slows down. In such cases, the process appears to converge to a limit, which is a characteristic of EAs.

An evolutionary process can be conceived as an optimization procedure; hence as a type of problem-solving, which tries to approach optimal values closer and closer through the migration of a species of solution candidates within a complex search space. In such cases, EC is considered within the category of parallel adaptive search (de Jong, 2006, p. 71). Individuals of an evolutionary process may be taken as representing sample points in a multi-dimensional search space that collectively migrate towards regions of high fitness (de Jong, 2006, p. 71). The evaluation function of an EA represents a heuristic estimation of solution quality (Eiben and Smith, 2003, p. 17). When the evolutionary process is terminated, the best point found can be viewed as the solution to the problem (de Jong, 2006, p. 71). In this regard, simple EAs can be viewed as a problem-independent paradigm for designing effective search procedures (de Jong, 2006, p. 72).

EC can also be envisaged as an adaptation process, where individuals compete with each other while adapting to an ever-changing environment. Matching environmental requirements ever more closely implies an increased viability, which is reflected in a higher number of offspring (Eiben and Smith, 2003, p. 16).

Many components of EAs are stochastic. Initiation, selection, mutation, and recombination operators may all operate in stochastic terms. For selection operators, although individuals with higher fitness levels have a higher chance to be selected, typically even the weak individuals have a chance to become parents or to survive. Likewise, for recombination and mutation, the choice of which pieces will be recombined or mutated is often randomized (Eiben and Smith, 2003, p. 16).

From a technical perspective, the utility of computational problem solvers is obvious. If there are repetitive problem types within a field, which involve a great amount of computation requiring the use of computers, entrusting these problems to automated systems would save human time and effort. Delivering a separate solution procedure for each such problem type might be a straightforward solution; however, if there are generic approaches that are applicable to a wide range of problem types, this would

be much more beneficial. Such approaches should not require much tailoring for specific problems, and should deliver good—although not necessarily optimal—solutions within acceptable time limits. EC provides an answer to this challenge (Eiben and Smith, 2003, p. 8).

The most dominant application area of EC has been optimization problems (de Jong, 206, p. 23). EC is used especially for problems that involve complex non-linear component interactions and when an algorithmic method to reach the optimum solution does not exist. However, utilization of EC is not limited to optimization problems. As flexible adaptive systems, EAs have found a range of applications varying from constraint satisfaction problems and economic modeling to simulation and the study of biological processes (Eiben and Smith, 2003, p. 13; de Jong, 2006, p. 105).

3.2 Brief History of Evolutionary Computation

The basic idea of viewing evolution as a computational process has been conceived in various forms during the first half of the twentieth century (de Jong, 2006, pp. 23-32). With the advent of cheaper computing machines, starting with the 1960s several groups became motivated by the idea that simple evolutionary models could be expressed in computational forms and could be used for complex computer based problem-solving. EC advanced within four main lineages. In each case, the EAs represented ideal models of evolutionary processes embedded in the larger context of a problem-solving paradigm. The key issue was identifying and capturing computationally useful aspects of evolutionary processes (de Jong, 2006, pp. 23-32). Amongst these four historical lineages, Evolution Strategies (ES) was invented in the early 1960s by Rechenberg and Schwefel, while working on an application concerning shape optimization (Eiben and Smith, 2003, p. 71). It was being used for difficult real-valued parameter optimization problems (de Jong, 2006, pp. 23-32).

The second lineage, i.e., Evolutionary Programming (EP) was originally developed to simulate evolution as a learning process with the aim of generating artificial intelligence. Intelligence was viewed as the capability of a system to adapt its behavior in order to meet some pre-specified goals within a range of environments; a view, which renders adaptive behavior as a key term (Eiben and Smith, 2003, p. 89). These ideas were initially explored in a context in which intelligent agents were

represented as finite state machines, and were proved effective in evolving better finite state machines over time (de Jong, 2006, pp. 23-32).

Genetic Algorithms (GA) were initially conceived by Holland as a means of studying adaptive behavior (Eiben and Smith, p. 2003, p. 37). Holland saw evolutionary processes as a key element in the design and implementation of robust adaptive systems that would be capable of dealing with an uncertain and changing environment, by self-adapting over time as a function of feedback obtained from the interaction with an environment. The early focus of GA was on developing more application-independent algorithms. The approach used a universal string representation for individuals, fitness proportionate selection, a low probability of mutation, and an emphasis on string-oriented reproductive operators for recombination and mutation (Eiben and Smith, 2003, p. 38; de Jong, 2006, pp. 23-32).

The above-mentioned lineages of EC have all originated during the 1960s. In contrast, Genetic Programming (GP), championed by Koza, appeared in the early 1990s. It differs from other EC strands in its usage of tree-based representations, and in its application area. While the older lineages of EC are mostly applied to optimization problems, GP can be positioned within machine-learning studies, as it is used to seek models with maximum fit. Modeling problems can be seen as special cases of optimization. In this case, models are treated as individuals, and their fitness is the model quality to be maximized (Eiben and Smith, 2006, p. 101).

During the 1970s, EC research focused on attempts to gain additional insight towards the core aspects of EAs through empirical studies and extensions to existing theory (de Jong, 2006, pp. 23-32). In 1980s, researchers started to use the already gathered set of algorithms as a basis for scaling up to problems that are more complex, and for developing new EC-based problem solvers for other problem domains (de Jong, 2006, pp. 23-32).

The emergence of various EC conferences in the late 1980s and in early 1990s enabled the representatives from various EC groups to meet each other. The result of the interactions was a better understanding of the similarities and differences of the various paradigms. The consequence of these meetings was a broadening of the perspectives of the various viewpoints. An array of new EAs were developed, none

of which fit nicely into the canonical EC paradigms, and it became clear that a broader, unified, and more fundamental view of the field was required (de Jong, 2006, pp. 23-32). Evolutionary Computation (EC) was agreed upon as the name of the field, and the traditional research lineages started to be taken as sub-fields of the same broad research field.

3.3 Evolutionary Computation in Art, Design, and Architecture

Evolutionary Computation approaches have already been tested in a diverse array of areas. This diverse array includes several design and art related fields as well. Indeed, EC has seen a wide variety of applications in arts and design fields. In design fields, a straightforward idea has been to use Genetic Algorithms (GAs) to optimize parametrically defined two or three-dimensional shapes. In these cases, the genotype most often consists of the parameters that govern a shape of a product and the evaluation is carried out over the phenotype representation. A seminal example of this approach is the evolutionary design of a boom that would hold a satellite dish (Keane and Brown, 1996). The strange shape of the resulting design, while better than the initial regular shape in terms of structural vibration control, was by no means expectable (Figure 3.1).

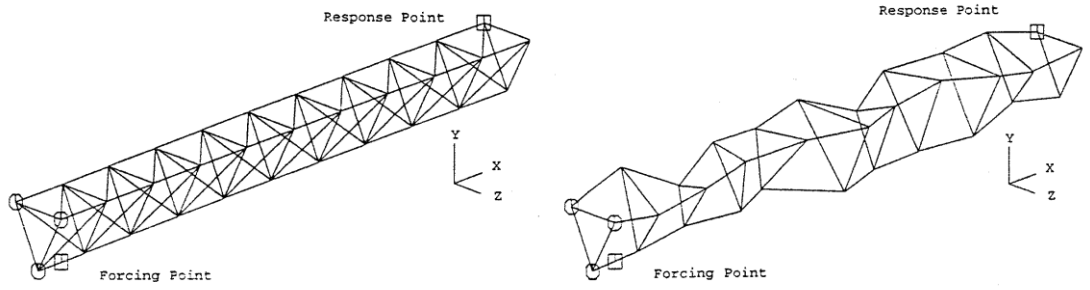


Figure 3.1 : Satellite dish holder boom; left, initial regular design; right, geometry of the final optimized design (Keane and Brown, 1996).

EC has been very popular in generative art circles; at least since, in his book "The Blind Watchmaker", Richard Dawkins described how evolution could be used to evolve shapes (Dawkins, 1996, pp. 43-74; Lewis, 2008). Following Dawkins, and the Genetic Programming (GP) variant, Karl Sims and William Latham evolved two-dimensional abstract illustrations in the early 1990s (Lewis, 2008). These studies were using tree-based mathematical expressions for genotype representation. This approach has been adopted in most subsequent evolutionary art. Karl Sims has also

experimented with artificial life[15], which became a research area on its own. In the succeeding decades, a generation of artists and researchers have recombined, modified, and extended these techniques[16] (Lewis, 2008).

Most evolutionary art systems utilize algorithms or mathematical expressions that yield a color value for a given canvas coordinate, thus defining a picture pixel by pixel. The tree-based mathematical expressions are used to generate regularities like curves, lines, and subtle gradients. Starting with initial expressions, systems continue by combining or modifying those expressions, so that offspring images carry some similarity to the initial patterns, but are still different.

There have been attempts to use evolution for generating music (see for example Bentley, 2002, pp. 163-250), for product design (Liu and Tang, 2006; Ang et al., 2006) and for generating two-dimensional forms and graphic layouts (Masui, 1994; Geigel and Loui, 2001; Ross, Ralph and Zong, 2006; da Silva Garza, Lores, and Zamora, 2008). Celestino Soddu (2002) developed a series of evolutionary computer applications, which he called, Basilica, Giotto, and Argenia, for generating images for cities, chairs, and buildings from initial DNA definitions that are developed by Soddu for each new problem.

There have been studies in architecture, by John Frazer, who has been a precursor of the usage of EC in architecture (Frazer, 1995). Frazer's long history of research on evolutionary architecture has focused on procedures for controlling growth and development from seed forms into emerging structures. His approaches are rooted in biological analogies, and they draw from a long list of generative and artificial life techniques (Frazer, 1995; Lewis, 2008). As examples, Frazer et al. (2002) demonstrated particle-based methods for building envelope design. Liu, Tang, and Frazer (2002) proposed an evolutionary multi-agent cooperative design environment. Gu, Tang, and Frazer (2006) experimented with capturing the intentions of a human designer during an interactive evolutionary process. Following Frazer's idea of "concept seeding", one of his students, Patrick Janssen (2004, 2006a, 2006b, 2009),

---

[15] For artificial creatures evolved by Sims, see Url-16.

[16] Although at any time a number of example EC-art systems can be browsed on the internet, these are mostly short-lived and non-maintained experiments. A re-implementation of the original "Biomorph" algorithm of Dawkins may be found in Url-17.

developed a framework for evolutionary design, which includes a "schema coding" stage that would be carried out by a designer team, to be followed by an automated evolutionary application phase.

As for rather well-defined problem areas of architecture, a series of experimentations have been carried out by Caldas, Norford, and Rocha (Caldas and Rocha 2001; Caldas and Norford, 2002, 2003; Caldas, 2003, 2005, 2006, 2008), which use EC for the aim of integral building envelope design and performance optimization. In these cases, given an existing building organization, shapes and heights of rooms and window sizes are optimized according to conflicting objectives like maximizing daylight and minimizing energy consumption. In a series of papers, the researchers presented examples of a continuous optimization workflow spanning over several design stages. Although these experimentations have been conducted for the latter phases of architectural design, where programming, design unit placement, and shape characteristics had largely been defined, they are important in showing how and where well-defined problems might be tackled by EC, in architectural design processes.

Mike Rosenman (1996, 2000) studied interactive evolution for floor plan generation. While the evaluation for some functional criteria was left to the computer, human designers were responsible for the evaluation of the formal qualities of candidate layouts. Rosenman and Saunders (2003) experimented with a self-regulatory hierarchical co-evolution model for designing.

John Gero's research group studied a diverse array of tasks through EC. They experimented with space layout topologies, combination of shape grammars with evolutionary approaches, and evolving linear plan units as design genes within a two-phased hierarchical evolution (Gero, Louis, and Kundu, 1994; Gero, Schnier, and Thorsten, 1995; Damski and Gero, 1997; Gero and Kazakov, 1998; Jo and Gero, 1998).

Several architects have been interested in EC as a generative tool, and developed specific design approaches. Chouchoulas and Day integrated shape grammars with EC for a generative architectural design approach (Chouchoulas, 2003; Chouchoulas and Day, 2007). Hemberg et al. (2008) experimented with a generative design tool called “Genr8”. Dillenburger et al. (2009) developed a complex design system where

conceptual building designs are generated using a series of desired characteristics for cells (or rooms) within the building site. Their fitness criteria include architectural qualities, which can be adjusted by parameter sliders.

With a performance-oriented design approach, Turrin, Buelow, and Stouffs developed a method to combine parametric modeling and genetic algorithms (Turrin et al., 2011; Turrin, von Buelow, and Stouffs, 2011). They present "ParaGen" as a tool, which combines parametric modeling, performance simulation, and GAs, together with a database to store and retrieve the solutions for subsequent exploration. ParaGen helps its user to automatically generate a range of alternative design solutions, and evaluate the resulting alternatives based on engineering criteria. The approach is demonstrated through the design of a large roof structure for a semi-outdoor space, taking day-lighting and thermal comfort issues as the main objectives.

Michael Bitterman (2010), proposed a cognitive approach for design performance evaluation. He developed methods to take experiential aspects of architectural spaces into account and used these methods in architectural layout studies on several scales.

Finally, in a series of interrelated studies, Sean Hanna (2005, 2006, 2007a, 2007b) utilized machine-learning techniques to generate implicit objective functions for usage in evolutionary generation of floor layouts. He used example layouts to train neural networks and employed these networks for the evaluation of candidate layouts in evolutionary generation tasks.

3.4 Basic Evolutionary Algorithm Issues

There are several basic components of an evolutionary system, which must be specified in order to create a working EA. Appropriate choices for each of these specifications are a function of both general, domain-independent issues and the specific properties of a particular application area (de Jong, 2006, p. 72).

In any EA, there will be one or more populations of individuals competing for limited resources; therefore, the first task is the definition of the individuals with an appropriate representation. Representation here amounts to specifying a mapping from the phenotypes onto a set of genotypes, which also sets up a bridge between the original problem context and the problem-solving space where evolution takes place. Objects forming possible solutions within the original problem context are referred to

as phenotypes, while their encodings, i.e., the individuals within the EA, are called genotypes (Eiben and Smith, 2003, p. 18).

A general technique for representation is to describe an individual as a fixed length vector of several features that are chosen because of their potential relevance in estimating an individual's fitness (de Jong, 2006, p. 3). The variables within the vector are used as parameters in the development of the phenotype. Another approach is to use tree-shaped representations, where the nodes hold a series of productions that, when applied, would produce a corresponding phenotype. The following are the core components of a simple evolutionary algorithm:

- Genotype / phenotype representations for the individuals
- Parent population
- Offspring population
- A set of evaluation methods
- Parent selection methods
- Survival selection methods (replacement)
- A set of reproductive (variation) operators (crossover and mutation)

Population size, population dynamics, and if necessary, additional structures have to be determined, in addition to the representation of individuals. The parent population size can be viewed as a measure of the degree of the parallel search an EA supports, since the parent population is the basis for generating new search points (de Jong, 2006, p. 50). Size of the parent population can be fixed or dynamic.

The evaluation approach forms the basis for selection. It is a function or procedure that assigns a quality measure to the individuals. It facilitates improvements, hence represents the requirements to adapt to (Eiben and Smith, 2003, p. 19).

In evolutionary computation, selection operators are responsible for pushing quality improvements. There are two basic stages for selection. The role of parent selection or mating selection is to distinguish among individuals based on their quality, in particular, to allow the better individuals to become parents of the next generation. An individual is a parent if it has been selected to undergo variation in order to create offspring. Parent selection is typically probabilistic to overcome the danger of the

search to become too greedy and to get stuck in local optima (Eiben and Smith, 2003, p. 20). Usually, after a set of offspring is created, another selection procedure is required for choosing individuals that will be allowed in the next generation. The role of survivor selection or environmental selection is to distinguish among individuals based on their quality. This decision is usually based on the fitness values of individuals, favoring those with higher quality, although the concept of age is also frequently used (Eiben and Smith, 2003, pp. 22-23).

The offspring population size plays different roles in an EA. One important role is related to the exploration-exploitation balance that is critical for good evolutionary search behavior. The number of offspring generated is a measure of how long one is willing to continue to use the current parent population as the basis for generating new offspring without integrating the newly generated high-fitness offspring back into the parent population (de Jong, 2006, p. 52).

The role of variation operators, or reproductive mechanisms, is to create new individuals from old ones. This amounts to the generation of new candidate solutions. From the generate-and-test search perspective, variation operators perform the 'generate' step.

The basic one-parent reproductive mechanism is mutation, which operates by cloning a parent and then providing some variation by modifying one or more genes in the offspring's genotype. The amount of variation is controlled by specifying how many genes are to be modified and how (de Jong, 2006, p. 61). The basic two-parent reproductive mechanism is recombination in which subcomponents of the parents' genotypes are cloned and reassembled to create an offspring genotype (de Jong, 2006, pp. 63-64).

Finally, an initiation procedure, a termination condition for stopping the evolutionary process, and mechanisms for returning the answers are required. In most EC applications, initiation of the first population is seeded by randomly generated individuals, although problem-specific heuristics can also be used with the aim of starting with an initial population with a higher average fitness (Eiben and Smith, 2003, p. 23).

If the problem has a known optimal fitness level, reaching this level should be used as the stopping condition. However, because EAs are stochastic and because most of

the times there are no guarantees to reach an optimum, this condition should be extended with an additional one that certainly stops the algorithm. This condition might be static such as a maximally allowed CPU time lapse, or a given number of generations. It can also be based on the monitoring of the process. For example, the process might terminate if the fitness improvement remains under a threshold value, or if the population diversity drops under a given threshold (Eiben and Smith, 2003, pp. 23-24).

3.5 Evolutionary Computation for Design: Requirements, Advantages, Drawbacks

For evolutionary applications in design fields, the following issues have to be taken into consideration:

- An appropriate representation for the solution proposals: This representation will include a genotype that encodes the most essential characteristics of the solution candidate, and a mapping procedure between this genotype and a phenotype that will be the solution.
- Reproductive operators: These operators modify the solutions towards promising areas of the search space. Domain-independent, standardized operators have been developed for this task. However, in a complex field such as architectural design, domain-specific strategies may be considered in relation to specific representations.
- Design goals and constraints: Evaluative procedures should be determined with respect to these goals and constraints. These procedures can be automated or interactive. If there are more than one evaluation criteria, approaches should be developed for incorporating these in the same evolutionary process.
- Appropriate process parameters: There is a multitude of domain-independent parameters that define the character of an EA. Fine-tuning these parameters is required to arrive at better fitness regions.

The possible diversity of the combination of answers to these questions makes EC a diverse field of exploration for design. As was stated earlier, EC appears as a variant within the search paradigm, but one that alleviates several problems of the search

paradigm in its basic forms. Most notably, utilization of stochastic methods eliminates the need for systematic search, which eases the computational burden in cases of combinatorial complexity. Another benefit is that, although substantial care has to be taken for representational issues, due to the flexibility and versatility of EC, strict formalization of a problem is not necessary for EC approaches. Evaluation in EC may be open-ended, as is exemplified in artificial life and interactive artistic evolution.

EAs can exploit domain-specific knowledge in its various mechanisms; not only in evaluation procedures, but also in representation, initiation, and in custom-made reproductive operators. Moreover, different types of EAs that utilize mechanisms like hierarchical organization (multiple populations, phases, or objectives) or co-evolutionary processes may enrich the arsenal of the designer. The advantages that render EC potentially useful for design applications can be summarized as follows:

- EC is inherently stochastic, flexible, and versatile.
- EC can be applied to problems where the optimal solution is not known.
- EA's are generic, they can be considered as engines to be plugged into any type of shape generation and evaluation procedure.
- EC is a popular scientific field and evolutionary approaches are well-studied.
- EC offers a possibility for multi-objective evaluation.
- EC involves a generate-and-test mechanism, which loosely resembles the basic schema of human design processes (see-move-see cycles).
- Procedural and evaluative knowledge can be incorporated throughout evolutionary processes. Rules and constraints can be used in both modification and evaluation.
- Possibilities for complex, multi-population, and hierarchical variants of EA's are unlimited. It is possible to devise hierarchical EA varieties, which would correspond to structures for design problems.

Regarding its operational characteristics, EC also has drawbacks for design. Heuristic local search, which encompasses most EC applications, is more suitable for task environments where a narrow range of objectives is sufficient. When the number

of objectives and requirements increases, maximum attained fitness for each objective tends to diminish due to an increase in the complexity of the problem. Whenever guided by an objective, evolutionary processes display an opportunistic character and try to evolve process parameters to maximally satisfy this objective using any means at hand. These means mostly comprise design elements whose characters are determined by the parameters kept within individual genotypes, which is a limited resource. If different objectives are satisfied by different parameters, or if there is a large parameter environment—enough for all the objectives—the number of objectives may be increased indefinitely. However, when the additional objectives are evaluated over the same set of parameters, these additional objectives start competing for the same resources to satisfy themselves. This results in a tendency to converge at a lower fitness level for each objective, even when the objectives are not explicitly conflicting.

There are several approaches for multi-objective evolution. The rank-based multi-objective evolution approach tends to develop individuals that are averagely fine on all objectives simultaneously. This type of evolution displays a convergent character. The alternative and popular option for multi-objective evolution is the Pareto-based approaches, in which a Pareto front of non-dominated candidates is sought. A candidate is said to dominate another candidate if it is better than the other on all the objectives. In current Pareto-based EAs, control mechanisms are employed for ensuring an evenly distributed Pareto-front. Therefore, a series of individuals, which may be better on just a few of the objectives while being unacceptably worse on several others are ensured to reach the final generations, which appears as a divergent evolution. Generating a set of individuals that are fine on all objectives is important for design tasks where solutions have to be acceptable on a number of criteria simultaneously. This means, in both rank-based and Pareto-based cases, the objectives will have to be satisfied simultaneously, and for most problem definitions there will emerge a limit for increasing the number of objectives.

However, it can be intuitively claimed that the same problem has to occur for human designers as well, since humans employ limited cognitive resources at any moment in design processes. As was claimed in the previous chapter, human designers appear to cope with this problem by using dynamic focusing and framing capabilities. This enables them to follow a limited number of criteria at any moment, giving varying

weights of importance to each of them. By rapidly alternating between conceptual frames and by focusing on different aspects of a context while constantly re-determining what is essential or inessential at any moment, humans appear to diminish their cognitive workload and carry out only a small number of evaluations at any moment. Throughout the design process, iterations of these partial and tentative evaluations gradually transform from a rather loose and permissive overall character to a more rigorous optimization attitude to give way to an integrated product. In the end, this rather unsystematic and chaotic progression appears to achieve sensitive, multifaceted, and well-integrated evaluations.

As was claimed earlier, such a process requires contextual appreciation, i.e., high-level general intelligence. In this respect, typical problems of AI apply to EC as well. Current EC does not alleviate problems of representation (i.e., the problem of grounding the EC representation into the real world and vice-versa), contextuality, and overall contextual interpretation. Consequently, most EC studies focus on well-defined optimization aspects of design processes, or try to treat design problems as optimization problems.

Taken in their simple forms, EAs do not involve intelligent mechanisms for problem formulation phases. They are not intelligent agents. They are rather versatile soft-computing environments, mostly taken as helpers in problem-solving. Here lies both the advantage and disadvantage of the evolutionary techniques. Natural evolution does not have an overall manager, or an overarching aim; it is a self-organized mechanism, which in the end generates complexity. Artificial evolution on the other hand, is given an overall aim, e.g., to solve a specific problem. It is guided by the manner of representation, variation mechanisms, and fitness evaluations. Artificial evolution is usually controlled with this limited set of inherent mechanisms.

On the other hand, EC is able to work rather blindly, by employing a limited amount of intelligence, which is its greatest advantage given the current state of AI. It is able to use a small number of overall objective definitions to produce viable solutions. However, this is at the same time the limitation of EC. Where there are no intelligent agents to guide a transformation process, evolution generates variety rather blindly, and only following some globally defined objectives. It mostly converges within sub-optimal areas and when the problem is complicated, may yield low-quality products. Even when there is only one objective at an evolutionary process, the effectivity of

EC depends on the level and sophistication of the embedded evaluative and executive intelligence. Better executive and evaluative mechanisms would yield better products. If it had access to high-level machine intelligence—or human intelligence through interactive evolution—an evolutionary process could behave more intelligently. A future remedy for such issues would be the development of dynamic decomposition and integration techniques that could generate tentative structures for design processes and tasks. In addition, methods would be required for dynamically generating complex hierarchical EAs that would correspond to these tentative problem structures.

On a strategic level, if an EC system employs a monitoring mechanism that is able to appreciate the progression of the states of the process, transformation of the process elements can be guided in a more sensitive manner. In such a case, the process will start to resemble "exploration", where intelligent agents steer the process with a strategic overall awareness. However, exploration is more akin to "intelligent design", and it should be contrasted with gratuitous natural evolution. This means, EC can be enhanced in an essential manner only with the incorporation of other intelligent technologies. At first sight, this makes EC indefinitely open to development; however, at some point in this development, the process will stop resembling evolution and will become something else, i.e., design. This is how continuity may be assumed between evolution and design; not within the mechanisms of evolution, or within hierarchical EAs, but with a gradual integration with other intelligent mechanisms.

As soft computing environments that can direct a transformation process with minimal intelligence, EAs can take part within complex agents, or within human-tool complexes, and can aid in finding solutions to readily developed problems at some certain stage of a design process. Although development of a problem definition, in principle, may also be subject to EC, as will be discussed in the following sections, it is hard to claim that EC is amongst the best options for this task.

The essential points in this account can be summarized as follows: Design problems are complex and the complexity of a problem impairs the success of EAs. As the number of objectives increases, which is inevitable in design, success per objective diminishes. Evolving the candidates longer is not a remedy, because evolution tends to converge at some point, i.e., it stops improving. The remedy appears as

dynamically breaking down and re-integrating the design process. This is only possible with an intelligence capable of appreciating a design context. In addition, better executive and evaluative mechanisms may improve the success of the EAs.

In brief, EC approaches require intelligent technologies or human intelligence for problem formulation tasks as EC does not appear well-suited for this task. EAs are not intelligent agents; they are soft-computing environments. Evolution operates mostly through random operators and with a minimum of intelligence, which is both its strength and disadvantage. Evolution is not intelligent design. However, EC may be used by intelligent agents, or together with intelligent technologies to carry out design. However, with the incorporation of a multifaceted intelligence, the process would stop being evolution and would better be called design.

The potential of EC lies at this point, because it can operate with minimal intelligence, it makes it possible to use limited, partial, or weak AI technologies for design problems. Additionally, it provides collaboration environments for humans and computers. Therefore, as a soft-computing environment, EC caters for the continuity of research. As will be demonstrated with the d_p.layout implementation in Chapter 4, EC is already capable of generating draft proposals for design tasks. These drafts help in the initial structuring of the design problems, which accelerates the process. If artificial systems are admitted within human design environments with the help of such technologies, researchers may gain an opportunity to develop better systems with the cooperation of human designers.

As a final remark, that a task is ill-defined is not an insurmountable problem for EC-based approaches. As will be illustrated with the d_p.layout system, an EC-based architectural design assistant may be developed in an open-ended way with the utilization of a flexible representation and a similarity-based evaluation mechanism. The resulting EC platform can be quickly adapted to a range of problems, none of which is implied within the initial framework or task definition.

In order to conceptualize the cooperation of humans, EC-based systems, and additional technologies, a generic task structure is given in Figure 3.2. The task structure develops the exploratory co-evolution model of Maher, Poon, and Boulanger (1996) in two basic ways: First, rather than assuming creativity, emergent behavior, or exploration on behalf of complex EAs, it locates complex EAs only one

level above the simple ones. Secondly, in addition to EC systems, separate intelligent agents are posited as necessary, and the exploratory character is assumed to appear only at this level—after the addition of the intelligent agents. In its totality, the task structure in Figure 3.2 may be interpreted as illustrating the required capabilities for a single autonomous evolutionary design agent. However, it is also possible to dismantle this structure towards collaborative models for practical collaboration of human agents and EC based soft-computing systems.

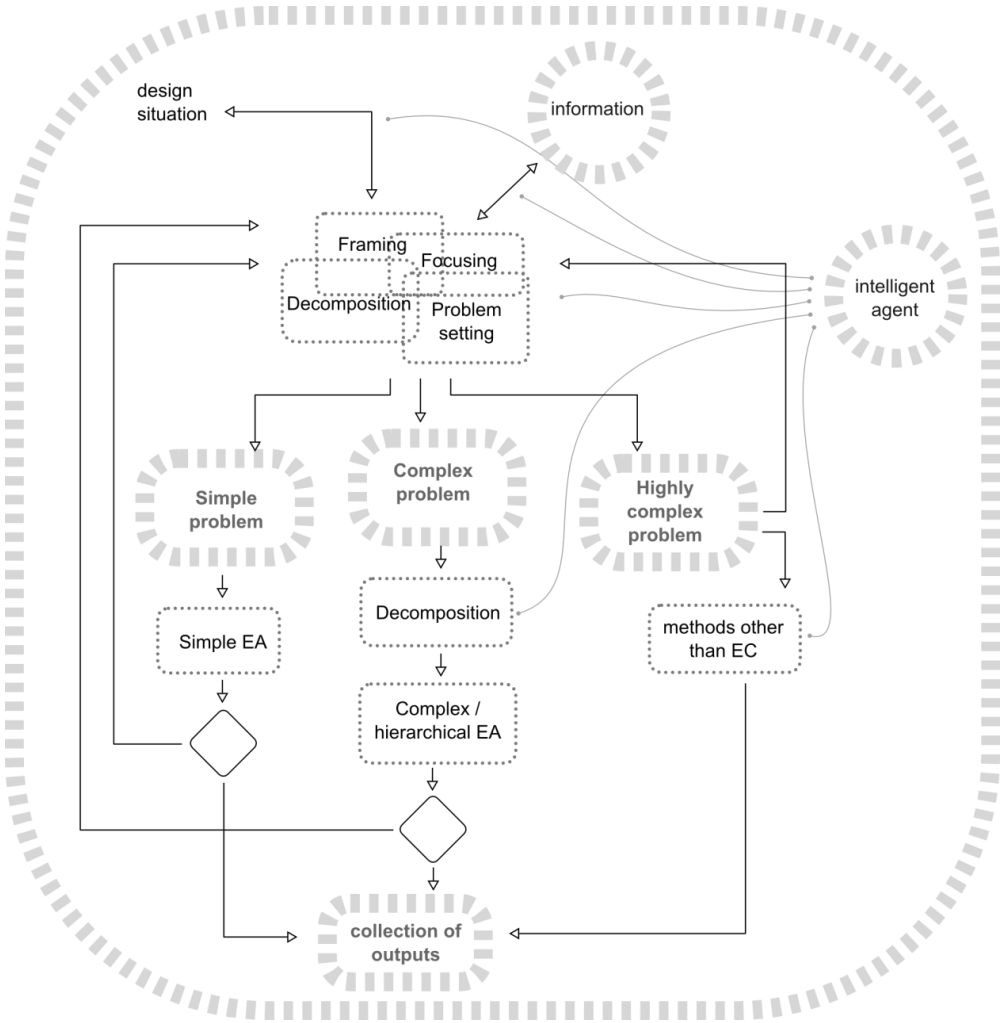


Figure 3.2 : An exploratory task structure for evolutionary design.

Three types of problems are distinguished within the task structure. As one of these types of problems is encountered by a guiding agent within a design process, this intelligent agent may assign the problem to an EC system, if possible and beneficial. For rather simple and well-studied tasks, such as optimization of the shape of a building component with respect to a small number of objectives, EC can be readily utilized. On the other hand, more complicated problems have to be either

decomposed into simpler ones, or structured in a way that they would correspond to a complex EA. If a problem is too complicated for EC, or if it would not benefit from EC aid, the agent may decide on using other means to tackle the problem. Further through the process, this may result in new problems that could be handled by EC systems. Consequently, the task structure illustrates a dynamic cycling of consecutive problem definition and solution processes.

In such a process, EC may be utilized whenever it is beneficial in some way, e.g., for problems that require excessive amount of computation. On the other hand, for conceptual phases of design, through the generation of rough drafts, EC may help in understanding a situation more comprehensively and faster, or help to make the design process more creative and pleasurable.

3.6 Complex and Hierarchical Evolutionary Algorithms

In this section, as potential evolutionary strategies for the task structure given in Figure 3.2, several options for complex hierarchical EAs will be examined. Because EC is an active field, new types of algorithms are constantly being developed and there is also a potential to develop design-specific EAs, as will be demonstrated by the Interleaved EA. Moreover, there can be an infinite variety of how separate evolutionary processes may be related, nested, and interacted within more complicated organizations. There are two basic requirements for incorporating such complex EAs in design. First, it should be possible to dynamically structure solution processes, and secondly, mechanisms are needed to develop complex EAs that would correspond to these problem structures.

In its various forms, problem structuring in terms of dynamic decomposition (i.e., focus) and integration is one of the basic strategies of human designers. The most obvious advantage of decomposition is a strategic reduction in the problem complexity. On the other hand, the obvious limitation for this strategy is the interdependence of the subtasks and productions in most design problems. This is why, human designers carry out decomposition in a highly dynamic and context-dependent manner and together with selective focusing mechanisms. Because strategies of decomposition are highly context-dependent, it is hard to obtain multi-purpose generalized decomposition strategies. Nevertheless, the following alternatives constitute abstract schemes for decomposition.

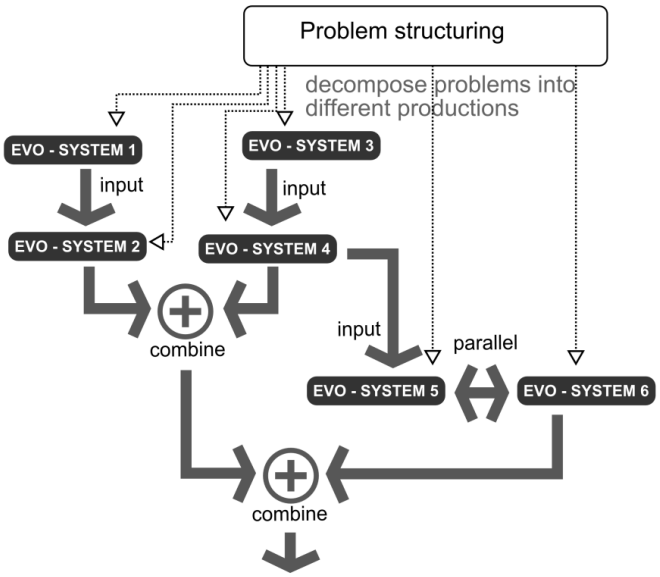


Figure 3.3 : Problem structuring (decomposition and integration) through productions.

In Figure 3.3, several possible paths are illustrated for problem structuring in terms of productions. Different evolutionary processes may be nested and combined, so that the output of one becomes the input for another. Another obvious strategy is developing different constituents of a product separately, to be combined later, such as parts of a chair or a table. Further strategies can be envisioned, where decomposed processes interact during mutual or parallel evolutions. In a similar manner, a structuring scheme may concern possibly simultaneous layers of a design process, as will be illustrated with layer-based decomposition and the Architectural Stem Cells Framework in Chapter 4.

A tightly integrated hierarchical strategy is adaptive evolution, where parallel evolutions are brought together into an integral evolution (Figure 3.4). In the simple case, the parameters that govern the evolutionary operators are evolved together with the products.

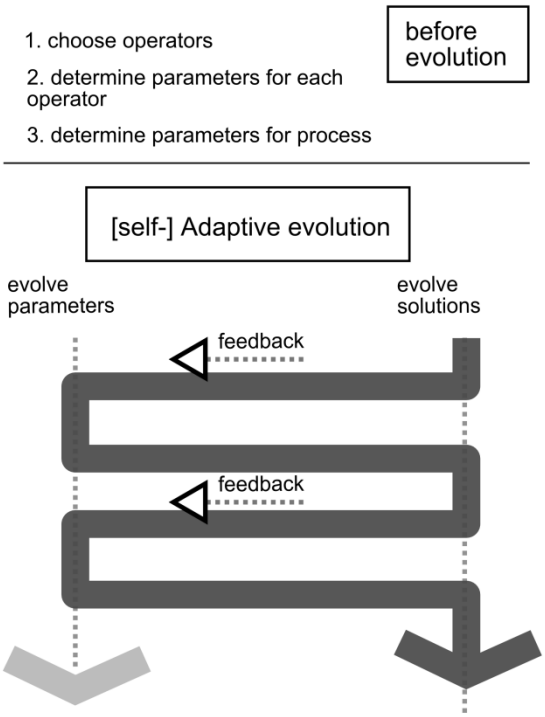


Figure 3.4 : Adaptive evolution.

There are several levels where adaptivity can be implemented. First, adaptivity may concern the whole process and use feedback from the health of the process. Secondly, to each candidate that is being evolved, its own operator parameters can be assigned. Thirdly, each component of each individual may be assigned its own parameters. Therefore, it can be claimed that in adaptive evolution, dynamic characters are assigned to each process, each candidate, or each component. In the last two cases, the feedback procedure is implicit. The underlying idea in these cases is that the operators and parameter settings that have created a better individual should be the better operators and settings. Therefore, although the overall mechanisms would still be fixed, the reproductive operators (crossover and mutation) can be co-evolved together with the products.

The idea of hierarchical evolution raises the possibility of evolutionary learning systems. As mentioned above with respect to Genetic Programming, EC has already been used for evolving machine-learning models. Such evolved models could be used for decision making during evolutionary design. It would be interesting to extend this idea towards a multi-level evolutionary approach, where the evolutionary processes compete and evolve. Such a system could function as an automated design system, which could generate and learn its own design methods.

On the level of parameters and settings, there are already various meta-level algorithms that can be used for tuning EA parameters for significant performance improvements with moderate computational costs (Eiben and Smit, 2012). An approach that has been used for such aims is "Racing". As the EAs are stochastic algorithms, performances of EAs are determined on the basis of a set of trials (usually 100 trials). In Racing, a fixed set of competing algorithms, or different instances of a single algorithm (each with different parameter configurations) are evaluated iteratively. This means that at each step all competitors are evaluated only once. After a sufficient number of trials, the distribution of the performance values of each competing algorithm starts to enable a comparison between algorithms. Through statistical methods, algorithms that are found to fall behind significantly are left out of the competition and the race continues with only the promising algorithms, which reduces the required experimental effort (Yuan and Gallagher, 2007).

Another approach is to use meta-EAs, in order to search for the optimal configuration for a specific EA. When there is a large number of parameters and settings, the number of the possible EA instances starts to increase combinatorially. In these cases, instead of using a set of fixed EA instances, the parameters can be evolved through a meta-EA (Yuan and Gallagher, 2007). It should be noted that, in this approach, although the population is evolving, at each evaluation step, there is a fixed set of competing algorithms, which enables the use of Racing methods, hence a hybridization of Racing and Meta-EAs (Yuan and Gallagher, 2007).

These parameter-tuning efforts should be considered the lowest level of the potential dynamism of the EAs. Indeed, a dynamism for a constant redefinition of the problem and solution strategies would be more beneficial from the viewpoint of design problems. Parameter settings may be thought to be redefining a solution strategy to a certain degree, yet the very definition of a problem and the evaluation approaches should also be evolving, and in real-time. Thus, in addition to the above-mentioned existing techniques, as an offshoot of artificial life studies, a possibility raised by the idea of hierarchical EAs will be shortly speculated upon below.

Within a universe of design agents and tools, a set of reusable actions and collections of these reusable actions may evolve autonomously as artificial life forms. A tool that is being used by an agent (in this context, parameters and settings) is thought to constitute a reusable action. These reusable actions are called "games" in this

context, to emphasize their reusable character. Figure 3.5 depicts such a scenario with three evolutionary levels. On the first level, a simple evolutionary algorithm is composed, using the items chosen from an inventory of agents and tools. This inventory may include simple procedures as well as complex applications or human agents. From this inventory, several tools and agents are selected to compose the games that will in turn function as the operators of an EA. Thus, each complete set of games will constitute an EA.

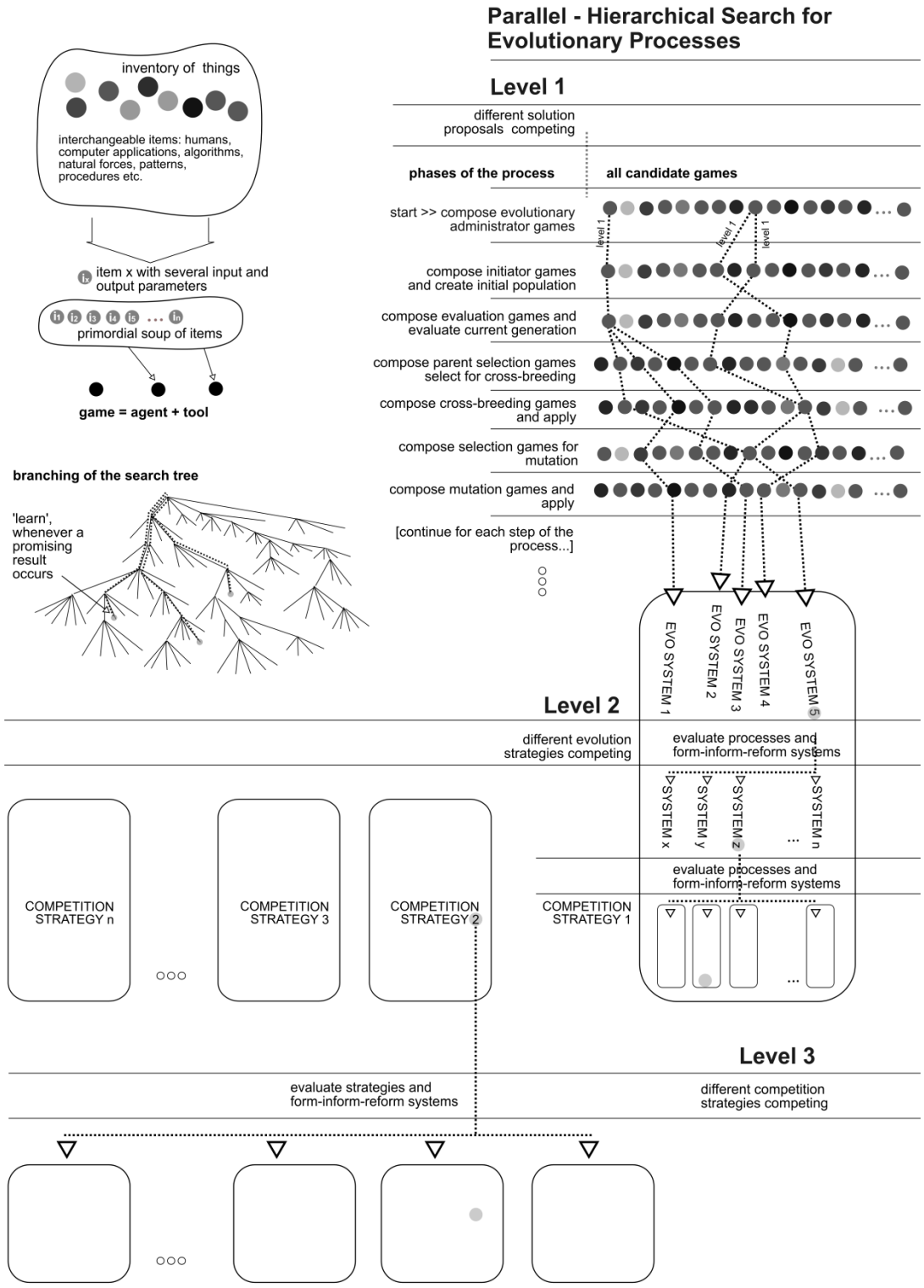


Figure 3.5 : Multi-level evolutionary search as a learning system.

On the second level, these alternative EAs tackle the same design problem; hence, they compete with each other. This second level can be envisaged as a meta-level EA that evolves EAs. At each generation, according to their success rates in coping with the design task, the algorithms will be assigned a fitness value, and they will be

reproduced by specific mutation and recombination operators. Recombination may function on the level of the operators of the EAs, thus EAs may exchange their games. As a more interesting possibility, throughout the evolutionary process, new games can be constituted and replace the existing games of the EAs as a mutation operation, which would provide a constant supply of new games. This would mean a nested evolution of the games as the basic contituents of the EAs, thereby the evolution of algorithms themselves in a dynamic manner. The evolution could continue indefinitely, as it has no inherent objective, except evolving life forms in a kind of arms race.

There may be various ways for comparing the success of competing EAs. Moreover, there may be various ways for composing games, mutating an EA, or for recombining several of them. Therefore, alternative evolutionary approaches can be generated for evolving EAs. Thus, a third level concerns such evolutionary processes where competition strategies evolve.

In theory, there are no limits for the number of higher or lower levels. However, it is obvious that, as a result of the combinatorial nature of this multi-level definition, and given limited computational resources, even when the branching factor is kept small, and even for just a few levels, such an approach will easily become impractical.

An unlimited number of potential variations of evolutionary processes can be envisaged. Compared with this unlimited search space, it seems practically impossible to autonomously develop more effective evolutionary algorithms with the described approach; hence, an unintelligent, mostly random, strictly evolutionary learning procedure for design methods is out of the question. Nevertheless, it is not compulsory to think of such an approach as fully evolutionary. An alternative conception may involve starting with tentative systems, with more human participation and gradually shifting weight towards automation. Human agents can carry out analyses over such a system (protocols of any process would be attainable) with the aim of detecting successful or unsuccessful design tendencies and patterns for the generation of more successful evolutionary design strategies.

Indeed, this multi-level evolution is reminiscent of what has been taking place within EC research. With the involvement of human agents on the first level, a multitude of productions have been carried out through a variety of EAs, which are constantly

being compared and tested on the second level. Research methodology over EC has also been improving, which can be compared to the third level in Figure 3.5, which develops competition strategies. The study of EC is the process of developing better EAs. Human researchers and other users of EC utilize their knowledge and intelligence for dynamic problem-setting and evaluation operations within this process, just as in a design process.

Domain-specific knowledge is an important aspect of design intelligence and it is retained both within the minds of human designers and within existing precedents or cases. Knowledge is required not only for reducing the complexity of design problems, or for structuring the design process for EC, but also for developing the evolutionary systems themselves. Although design knowledge is predominantly developed and applied by human designers, knowledge can be embedded within an EA as well. Methods for embedding knowledge into EAs may ease the burden of human users and may extend the scope of EC in design. There are several slots in an EA where knowledge can be inserted, most importantly, representation of the candidates, mutation operators, and evaluation procedures. Although usually kept generic, other operators such as selection and recombination operators may be used for domain-specific interventions as well.

EC offers generic methods for most of its operations. Indeed, when taken as a versatile multi-purpose problem-solving approach, this generic manner appears to be the essence of EC. However, design problems are highly complicated and domain-specific knowledge and skills appear necessary to structure and solve these problems. In EC, utilization of domain specific knowledge has the potential to convert a generic search approach into a highly constrained, domain-specific, heuristic search. However, it should be remembered that, in real world problems domain specific knowledge and skills have to be employed together with general daily intelligence. Therefore, accumulation of such knowledge may answer only one side of the problem of design intelligence. This issue will be revisited within the context of the Architectural Stem Cells approach in the next chapter.

3.7 Interleaved EA: Multi-Objective Evolutionary Computation for Design

In most design tasks there is a multitude of objectives to be met, therefore, a convenient multi-objective generative mechanism is required in order to tackle them.

This section will present a complex, multi-objective evolutionary algorithm, which is called the Interleaved EA. The specificity of the Interleaved EA is that one of the objectives leads the evolution until its fitness progression stagnates, in the sense that the settings and fitness values of this objective is used for most evolutionary decisions. This may be called the leading objective principle. In this way, the Interleaved EA enables the use of different settings and operators for each of the objectives within an overall task, which would be the same for all objectives in a regular multi-objective EA. Such a setting also enables the use of some of the single-objective EA mechanisms in multi-objective evolution.

The Interleaved EA offers an improvable method for the utilization of domain-specific knowledge in EC. It has been developed with respect to an analogy with how a human designer works, who focuses temporarily on a limited set of aspects, and after improving on that temporary set, moves forward to improve another group of aspects, which was possibly degraded by the previous operations. As such, the Interleaved EA also demonstrates where new intelligent technologies could be inserted, in other words, it is essentially open to further development through additional methods to determine when to follow which objective and with which mutation operators.

Interleaved EA is a variant of the "criterion-based" methods for multi-objective optimization, which switch between the objectives during selection phases. At each selection stage, potentially different objectives decide which members of the population will be selected (Zitzler, Laumanns, and Bleuler, 2004). Sometimes a probability is assigned to each objective, which determines whether the objective will be the sorting criterion in the next selection step. In the lexicographic approach, which is adapted for the naive version of the Interleaved EA, the objectives are assigned priorities before optimization and the objective with the highest priority is used first when comparing individuals in a single-objective manner (Back, Fogel, and Michalewicz, 2000).

In the Interleaved EA, each objective guides the evolution with its settings and operators until the fitness improvement stagnates for this objective (Figure 3.6). It should be noted that a minimum number of generations is assigned to each new leading objective in the beginning of its lead, whether it stagnates or not. In practice, this is mostly a number between 8 and 32. Thus, the Interleaved EA is differentiated

from the earlier approaches with a dedicated period for the domination of each objective, which is dependent on the feedback from the progression of the fitness values. This behavior of the algorithm is further exploited to enable the use of separate operators and settings for each of the objectives (Figure 3.7). Therefore the main specialties of the Interleaved EA are:

1. Letting each objective govern the process until fitness improvement stagnates (Figure 3.6) (whenever this occurs, the lead is given to another objective).
2. Enabling the use of separate operators, parameters, and settings for each objective (Figure 3.7).

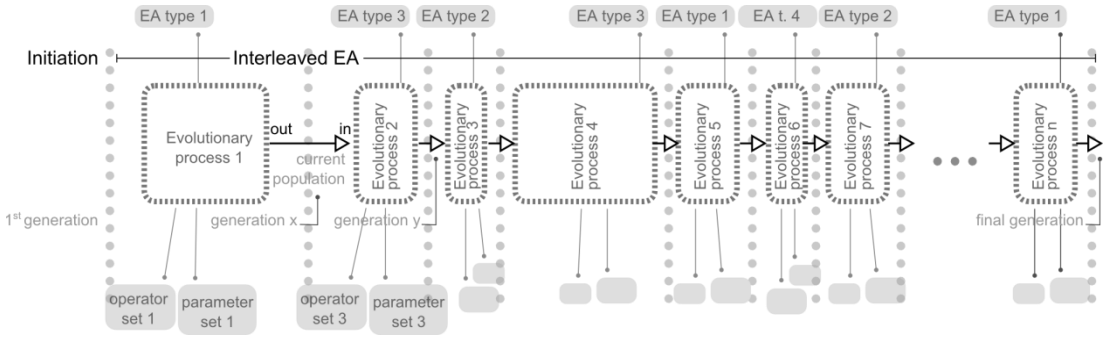


Figure 3.6 : Schematic illustration of an Interleaved EA process.

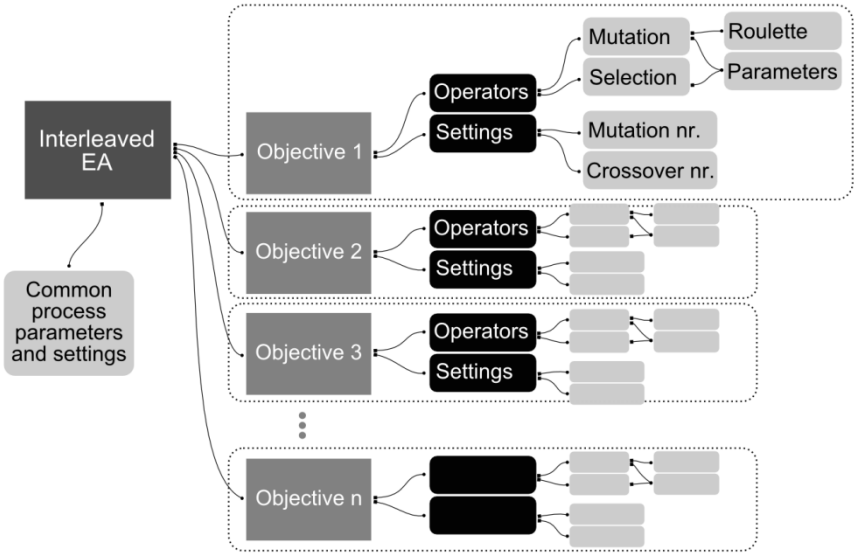


Figure 3.7 : Basic structure of the Interleaved EA.

The separation of operators and settings becomes meaningful with the use of domain-specific knowledge. When it is suspected that the separate objectives may require different operators (mutation and selection operators) and could respond to

different evolutionary settings (mutation and crossover numbers), and parameters (roulette parameters, mutation ratios, step sizes, etc.) the separation of operators, parameters, and settings may be beneficial (as in the obvious case of the graphic application in Chapter 4, where the main mutation operator for the "Color" objective is different from the "Cell" and "Sequence" objectives). This separation also enables, in the adaptive cases, a separate adaptation of the parameters of each objective, which may differ throughout the process. As a second benefit, this separation enables the user to adjust the objectives in terms of preferences. For example, through the mutation and crossover numbers, each objective may be given a more or less greedy setting, which can prioritize it over the others.

There are two versions of the Interleaved EA. In its naive version, the leading objective decides on both variation selections and the selection of the next generation. This version may rather be described as a dynamic series of single-objective EAs, operating successively over the same population. Each single-objective micro-evolution transforms the population until an adaptive average fitness threshold is reached and then leaves the population to another micro evolution, to be guided by the second objective on the priority list.

The naive version has been shown to work for the d_p.graphics application (Chapter 4), because although somewhat interfering, the objectives are not entirely conflicting in that case. However, this naive approach does not work for the architectural layout task (in Chapter 4), for which a rank-based multi-objective selection approach was required.

In the rank-based version of the Interleaved EA, variation selections and operations may still be carried out with respect to the leading objective; however, the selection of the new population is carried out using all of the objectives, with a rank-based selection operator. As such, this version is a proper multi-objective EA. Additionally, the priority list is disabled, and the next objective is determined randomly.

Within a complex EA, a large amount of process parameters have to be fine-tuned for the evolutionary process to perform well. Before the final evolutionary runs, a series of test runs can be carried out to obtain fine-tuned parameter combinations. However, these parameters are not independent from each other. This means that the parameters cannot be searched for separately. Instead, combinations of parameters

have to be tested. Moreover, a series of tests are required for each combination, because of the stochastic nature of EC. As a result, searching for the best values for all different combinations is a difficult problem. Some techniques for tackling this problem were mentioned in section 3.6. In any case, for complicated problems, the parameter tuning process can become time consuming even when parameters are optimized one by one regardless of their interactions. Furthermore, the best value combinations tend to vary for each different design scenario, preventing the development of a generic set of parameters, hence impairing the versatility of an algorithm. Even worse, the best values for parameters tend to change during the evolutionary process.

Regarding the parameter adjustment problem, Eiben and Smith (2003) make a distinction between "parameter tuning" and "parameter control". The former amounts to finding good values for the parameters before running the algorithm, which remain fixed during the run. The latter offers an alternative, as it suggests starting a run with initial parameter values that are to be adjusted during the run. Parameter control is further classified into three categories (Eiben and Smith, 2003):

1. Deterministic parameter control takes place when the value of a strategy parameter is altered by some predetermined way without using any feedback from the process.
2. Adaptive parameter control involves some form of feedback from the evolutionary process to determine the direction or magnitude of the change in a parameter. The parameters belong to the evolutionary process, thus, this case will be referred to as "process-based" adaptivity.
3. In "self-adaptive" parameter control, adaptive parameters are encoded into the chromosomes of the individuals and undergo mutation and recombination together with the individuals that they belong to—with the hope that the process finds the best values itself during the process.

The second case is inherent to the naive version of the Interleaved EA. There are several process parameters like a priority list for the objectives and fitness thresholds for each objective, which are adaptively altered during an evolutionary run. A self-adaptive variant has also been implemented for important mutation parameters, which is found successful, at least for the d_p.graphics application (Chapter 4). As

was discussed within the context of complex EAs, self-adaptive EAs are integral parallel evolutionary systems, where the evolution of the parameters is integrated within the evolution of the solution candidates. In this way, the feedback between the two evolutions is immediate, and much more complex than it could have been with a manually constituted static scheme. This co-evolution model of the Interleaved EA concerns the adaptation of complex EAs; in other words, it is a dynamic automated mechanism for complex EA adaptation. However, this is an implicit and lower level dynamism, and concerns the solution of a single, predefined task while complicated design fields demand the parallel evolution of a series of possibly interacting tasks, and even the task definitions. The minimum requirement for this demand is a higher-level hierarchy and parallelization on behalf of the evolutionary system as was discussed in the previous sections.

Figure 3.8 depicts a scenario for a complex parallelization and nesting of a set of design_proxy-based assistants. It should be remembered that the Interleaved EA has been mainly developed as the engine of the design_proxy-based assistants. It is a response to the fourth tenet of the design_proxy approach ("a dynamic evolutionary approach for solution generation") and it is expected to form an integrated whole with the methods that follow the other tenets[17]. In a multi-objective case, the similarity-based evaluation approach will contribute a set of objectives, to be used together with the other technical or formal objectives. As such, there will be a set of objectives with their operators and settings, each objective constituting a package. Each grouping of such objective packages towards a design task can constitute a separate design_proxy.x assistant (e.g. design_proxy.layout, design_proxy.facade, etc.). Finally, a set of design_proxy assistants that are layered over each other in a parallel evolution may constitute a higher-level design_proxy assistant (e.g., design_proxy.architecture).

[17] The design_proxy approach consists of, (1) flexible and relaxed task definitions and representations, (2) intuitive interfaces that make use of usual design media, (3) evaluation of solution proposals through their similarity to given examples, (4) a dynamic evolutionary approach for solution generation. See Chapter 2 for further information.

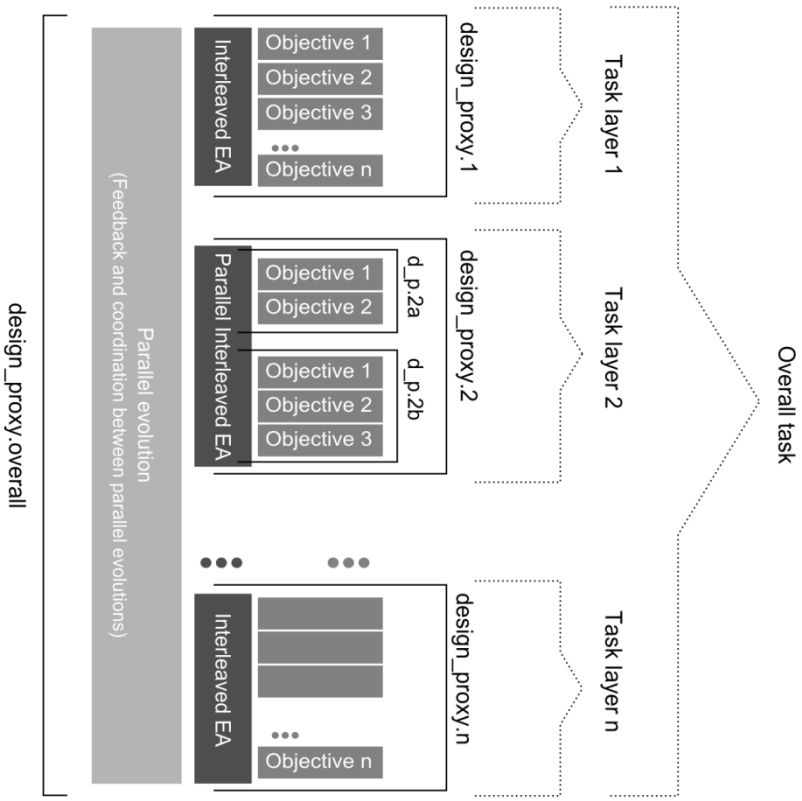


Figure 3.8 : Potential parallelization of design_proxy-based assistants.

3.8 Conclusion

This chapter concerned theoretical and technical issues regarding Evolutionary Computation (EC). In the beginning of the chapter, an overall introduction to the basics and the history of EC studies were given, and the basic issues, requirements, advantages, and drawbacks of EC were discussed. Additionally, as part of the conceptual background for the Interleaved EA, several schemes, illustrating complex and hierarchical EAs were presented and discussed.

The question concerning the required collaboration scheme for the utilization of EC within design tasks was answered by a generic task structure (Figure 3.2). The task structure locates complex EAs one step above the simpler ones, yet posits separate intelligent agents as necessary for the guiding of the process. A multi-level learning EC is hypothesized (Figure 3.5) during the discussion of the complex EAs, which would gather all its constituents within hierarchical levels of evolution.

These discussions culminated in the proposal of the "Interleaved Evolutionary Algorithm" (Interleaved EA, IEA). The Interleaved EA is a dynamic, adaptive, and multi-objective EA, in which one of the objectives leads the evolution until its fitness

progression stagnates; in the sense that the settings and fitness values of this objective is used for most evolutionary decisions. In this way, the Interleaved EA enables the use of different settings and operators for each of the objectives under an overall task, which would be the same for all objectives in a regular multi-objective EA. This property gives the algorithm a modular structure, which offers an improvable method for the utilization of domain-specific knowledge for each sub-task, i.e., objective.

The Interleaved EA has mainly been developed as the engine of the design_proxy-based assistants. It is a response to the requirement for a dynamic evolutionary approach for solution generation. With its modular structure, it may also be the basis for a multi-level, dynamic, parallel EA, for a higher-level design_proxy assistant.

4. APPLICATIONS OF THE DESIGN PROXY APPROACH AND THE INTERLEAVED EA

In this chapter, the design_proxy approach will be applied to two design tasks, i.e., the d_p.graphics and the d_p.layout. In these applications, the multi-objective evolution approach, adaptivity, and the evolutionary process will be attributed to the Interleaved EA, while the aspects of the specific application, i.e., task definition, representation, initiation, evaluation, variation methods, and the interface will be collected under the specific instance of the design_proxy.

4.1 design_proxy for Graphics

4.1.1 Task definition

The graphic task for developing and demonstrating Interleaved EA and d_p.graphics is defined as generating graphical arrangements using a canvas, basic design units (DUs, i.e., pattern stamps and text strings), and color sources (radial gradients and plain colors) (Figure 4.1). The aim is to generate graphical arrangements that resemble a series of target images, according to different characteristics—in this case, color distribution and graphic layout.

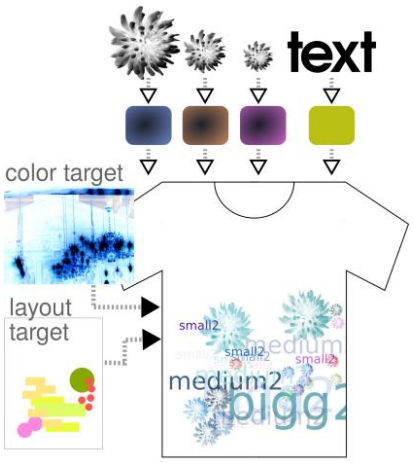


Figure 4.1 : Graphic arrangement task.

As can be remembered, the main tenets of the design_proxy approach are a relaxed problem definition, intuitive interface, similarity-based evaluation, and dynamic

generation. Because the graphic task is devised for the purpose of the development and testing of design_proxy and Interleaved EA, the task already involves all tenets of the design_proxy approach implicitly in its definition. The application will also be used for the initial illustration of several concepts that are brought forward or mentioned in this thesis.

While various tasks can be carried out within this definition, the series of applications for the following experimentations concern the generation of decorative graphic patterns. This toy problem involves two distinct sub-tasks, i.e., layout arrangement and color distribution, which require a set of variation operators and evaluation procedures for attaining these visual characteristics. Three main task-specific themes of this implementation will be:

1. The multi-objective EC approach.
2. Parameter setting procedures for EA parameters.
3. Methods for capturing human visual preferences.

Interleaved EA is the answer to the first issue and adaptive parameter adjustment is proposed as an answer to the second. Considering the third issue, the main reason behind the attempt to capture visual preferences within artificial constructs is the aim to automate the visual evaluation task. In general, automating a visual evaluation process is not a simple task, as visual characteristics, style, and taste are hard to define or to parameterize. A mechanism that aims at capturing aesthetic intentions cannot be static. Rather, it should be possible to define preferences on the run, with respect to the specific state of a design process.

A solution proposal for a graphical task is determined by several visual aspects, such as color and graphic arrangement. Each of these aspects of a desired solution can be defined by separate stylistic constructs and the desired condition for a solution may be defined by a combination of these separate stylistic constructs. For the definition of these preferences, instead of parametric or interactive approaches, a simple method is proposed to automatically evaluate the candidates. Desired visual characteristics are given to the system by means of target images. Images create a fast, intuitive, and controlled interface for the human designer, solving the evaluation and interface problems with one move. Thus a need for numeric parameter setting is eliminated; which could become cumbersome in complicated scenarios.

Within the application, each stylistic preference is defined with a target image that is supplied by a human user. Methods have been developed for measuring the resemblance between target and solution images with respect to the desired features. This way, each stylistic preference is measured in terms of its similarity to a paradigm case[18].

Within the generic evolutionary task structure in Figure 3.2, the above description of the graphic task corresponds to the problem-setting stage. It should be noted that this task definition constructs a prefabricated strategy, which can be used in different scenarios, according to the needs of different contexts. Again, with reference to the evolutionary task structure (Figure 3.2), the Interleaved EA is an example of the complex, hierarchical EAs.

4.1.2 A review of related studies

The above-described task, i.e., the generation of graphical arrangements using DUs, constitutes a sub-region of a broader layout generation problem. Layout generation is a long-studied field of Computational Design, which has many variants such as artistic illustration, graphic design, user interface design, architectural layout design, electronic circuit board layout, and space allocation. In design fields, computational layout generation methods usually involve distinct design units (DUs) to be arranged over a two-dimensional space according to a multitude of requirements.

Constraint-based approaches are common in visual interface layout generation, while template and grid-based layout organizers have also been utilized (Lok and Feiner, 2001). Lok, Feiner, and Ngai (2004) proposed automated layout generation based on automated evaluation of visual balance, while Geigel and Loui (2001) described a flexible system for automatic page layout that makes use of genetic algorithms for albuming applications. In the latter study, layout fitness measurement is based on graphic design preferences supplied by the user according to several criteria such as balance, spacing, chronology, emphasis, and unity.

---

[18] The importance of resemblance-based evaluation has been stressed by Dreyfus (1972; Dreyfus and Dreyfus, 1986) with reference to Wittgenstein's (1999) "family resemblance" notion, as an alternative to the rule and definition based approaches. Conceptual background and practical utility of this conception will be further elaborated in the next chapter.

Another line of research, which is closely related to this thesis, formulated the architectural layout problem as an evolutionary search. Hanna (2005, 2006, 2007a, 2007b) proposed to derive an objective function implicitly from a set of target spatial distributions using axial maps, and carried out experimentations on the representation of style. A more extensive review of the architectural layout problem will be given in the next chapter.

An image-based user interface for knowledge input for an architectural design context is exemplified in Dillenburger, Braach, and Hovestadt (2009). In a broader perspective, style extraction from given images may provide us with a huge style-base implicit in existing graphics. For the extraction task, Content Based Image Retrieval (CBIR) studies offer a wealth of fast and well-studied techniques (Smeulders et al, 2000; Datta et al, 2008; Veltkamp and Tanase, 2002). For the d_p applications, variants of techniques are developed and adapted from Smith and Chang (1999).

An automated visual evaluation method that is able to take user desires into consideration would be indispensable for an intelligent, automated, or semi-automated design system. While attempts have been made for automated visual evaluation in artistic illustration studies (Lewis, 2008), in design computation visual evaluation—if not completely omitted or hidden beneath functional performance—is still mostly relegated to the human judges, who are summoned either during or at the end of the processes. Interactive evolutionary algorithms exemplify one case, and parametric approaches another. In the latter case, parameters are adjusted before the process. After evaluation, the user is supposed to re-adjust the parameters and restart the process until desired results are obtained. Real-time versions can also be envisioned; however, in any case defining visual characteristics by numeric parameters can become a very complex and tedious task, while interactive sessions tend to become tiresome for the human judges.

A series of studies explored a neural net approach to learn the user preferences from the images evolved during interactive sessions. The learned information is then used to automate the evaluation stage (Baluja, Pomerleau, and Jochem, 1994; Wiens and Ross, 2002; Hewgill and Ross, 2004). Efforts have been made by Machado et al (2004, 2005) to use complexity estimates for images to produce automated image "critics" with the aim of using this knowledge for automated and semi-automated

image generation sessions. Ross, Ralph, and Zong (2006) attempted to automatically evolve procedural texture formulas by matching the candidates with artistic imagery. In the generation phase, a multi-objective Pareto optimization process is employed, where candidate textures are evaluated according to several objectives including bell curve distributions of color gradients and color histogram scoring by matching a candidate texture's color composition with the color histogram of a target image. The approaches proposed in this last study are precursors of the image-based definition procedures that are used for the d_p.graphics.

4.1.3 Visual interface

As already mentioned above, for the d_p.graphics application, an image-based interface is used to set the desired stylistic characteristics before an evolutionary process starts. A series of target images are used for this task. Through these images, separate target features for 2D arrangements are determined, which describe color distributions and spatial arrangements. This separation makes it possible to create novel graphical arrangements using different targets for each feature type (Figure 4.1). Three usages are proposed for the extracted features:

1. Initiation of the first generation (probability distribution maps are produced from target images).
2. Measuring color similarity between the target and candidate color distributions (features are defined by one-dimensional histograms).
3. Comparing target and candidate layouts in terms of DU sequences and distributions (grid-based layout evaluation).

Two basic types of target images are utilized for the definition of the preferences:

1. For color distributions, arbitrary bitmap images are supplied by the users.
2. For DU layouts, vector graphics are generated by the users with pre-specified DU types. These vector graphics are parsed with a script, so that coordinates of all the DUs are acquired with respect to DU type.

An initiation map is produced from this information. For the initiation map, each DU’s center coordinates are divided by a cell width. Row and column numbers of resulting cells are placed in a dictionary, which holds possible initiation positions for each DU type.

A number of procedures that measure a candidate image's similarity to the target images are used to capture and implement functional and aesthetic intentions of the users. These procedures act as types of objective functions, along with other formal or functional objective functions and constraints. Preparation of target features and the evaluation procedures will be described in the following sections.

4.1.4 Representation, initiation, evaluation, operators, and the evolutionary process

In the d_p.graphics implementation, the genotype of a candidate image comprises all the data related to that candidate, including phenotype mapping information (Figure 4.2). Both evolved and fixed values are stored in an individual's genotype. Each DU has a static location on a fixed length chromosome. For each DU, evolved values comprise the x and y coordinates of the DU center and RGB values for the gradient stops.

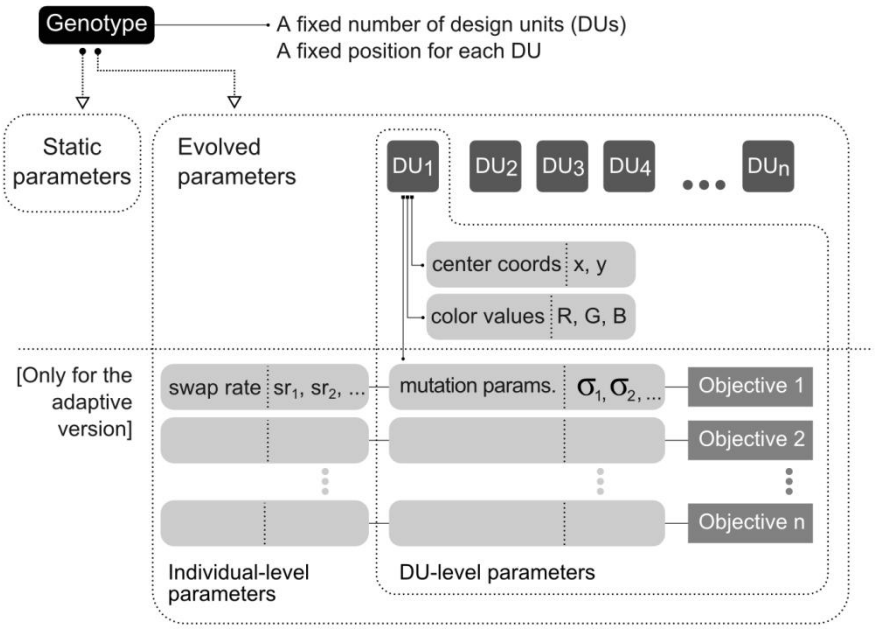


Figure 4.2 : Genotype for d_p.graphics.

In the self-adaptive version, for each objective function, separate lists of standard deviations for 'nudge' and 'color' mutations, as well as 'swap rate' values are also stored and evolved as follows:

- For each DU, one standard deviation value is stored, for each of the nudge and color mutations (these mutations will be described below). This is an example of component-level self-adaptivity.
- For each candidate (i.e., individual) a value is stored as the ‘swap rate’. This exemplifies individual-level self-adaptivity.

A candidate is initiated by assigning center coordinates, color values, and optional mutation parameters to all of its DUs. Center coordinates are initiated either from amongst possible values of the initiation map or randomly (Figure 4.3).

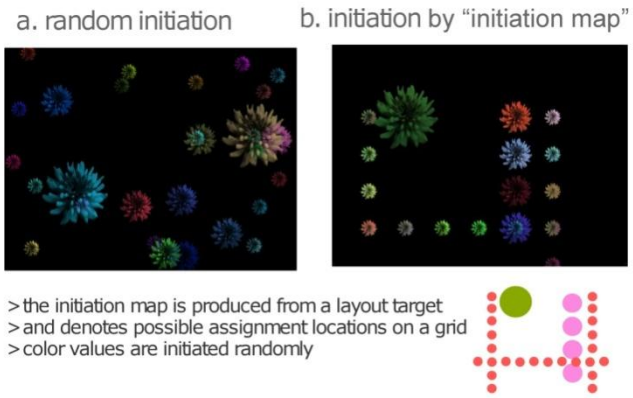


Figure 4.3 : Initiation (Desktop 1 scenario).

In total, three selection operators are implemented:

1. Fitness proportional selection: In the first step, fitness values of the candidates are normalized, i.e., the fitness value of the worst candidate is subtracted from the fitness values of all candidates, then all values are divided by the sum of all fitnesses, so that the worst candidate is assigned a value of 0, and the total value of fitnesses is 1. In the second step, for each candidate, a random number is drawn from a uniform distribution, within the range of [0-1]. If this number is below the normalized fitness of the candidate, the candidate is selected. This procedure is repeated until the desired amount of candidates is selected. This method assigns a probability of selection for all candidates except the worst one. At the mean time, a small advantage is provided for better candidates. Additionally, a pre-specified number of best individuals can be selected automatically. If the sum of all fitness values is zero, the selection is carried out in a uniform manner.

2. Tournament selection: In the typical case, a selection tournament concerns two candidates; however, tournaments amongst more than two candidates are also possible. This is sometimes preferred for a stronger selection pressure. Therefore, in the implemented tournament selection mechanism, it is possible to carry out tournaments amongst a multitude of candidates. The number of candidates can be as large as the selection pool. In the first step, a pre-specified number of candidates are selected amongst all candidates—with replacement, to prevent a candidate to compete with itself. The tournament procedure is straightforward. The best individual is selected amongst the tournament candidates. Additionally, a pre-specified number of best individuals can be selected automatically, without entering into a tournament.
3. Uniform selection: This is a random selection procedure amongst several candidates. Additionally, a pre-specified number of best individuals can be selected automatically.

There are several stages in an EA, where selection pressure can be applied. These are the stages where crossover parents, mutation candidates, and individuals for the next generation are selected. For controlling the speed and progression of an EA, and most importantly for avoiding greedy EAs, which can get stuck in local optima, the selection pressure should be balanced. Therefore, it is sometimes desirable to carry out some phases of selections with no pressure at all, while applying pressure at other phases. There are also several parameters that control selection pressure, such as the quantity of candidates to be generated by recombination or mutation. Other parameters that are used for the other operators may also interfere with the selection pressure. Therefore, it is not possible to decide on these parameters in an a priori manner and extensive tests have to be carried out for finding appropriate value combinations.

For recombination of parent candidates, an ‘n point crossover’ operator is implemented. It is a procedure where a series of randomly selected DUs are exchanged between two genotypes. When the DU positions are static on the implemented genotype, it is straightforward to implement this process. Within the self-adaptive version, several parameters are attached to each DU, and these are transferred together with the DU. However, there are also adaptive parameters

attached to a candidate, i.e., adaptive sigma rates and swap ratios. These values are not exchanged.

There are three mutation operators (Figure 4.4):

1. Nudge mutation is a type of "creep mutation" (Eiben and Smith, 2003). For each DU of a candidate, first, a number is drawn for the nudge step from a Gaussian distribution with mean 0, and standard deviation σ. If the unsigned value of this number is below a specified threshold, it is ignored; else, according to another random draw, it is summed up with the present x and/or y value of the DU. This latter procedure is also a means to limit the number and strength of mutations for a candidate. In the self-adaptive version, for each fitness type, a separate standard deviation value is stored in each of the DUs. These values are again mutated with a creep mutation operator, before the nudge mutations are carried out.

2. For "swap mutation", a candidate is chosen, and a random number is drawn within the range [0-1], if this number is below a certain swap threshold, a set of DU couples are selected from the individual's chromosome. The number of these couples is controlled by the "swap rate" which is assigned to each candidate. In the next step, the center coordinates of these DU couples are exchanged. The swap threshold controls the amount of swap mutations within a population; however, because this is also controlled by the "swap rate" which can be an adaptive parameter, the former threshold is eliminated by being set to 1. In the adaptive version, for each candidate and for each fitness type separate swap rates are stored. Swap rate values are bounded at 0 and 1 and mutated using a creep mutation before the swap mutations are applied.

3. "Color mutation" is implemented as follows: First, the RGB values are converted into HSV mode. These values are in turn normalized within the range [0-1]. After this step, for each of the H, S, and V values, the value is mutated by a creep mutation, similar to the nudge mutation above. A number is drawn for the mutation step, from a Gaussian distribution with mean 0, and standard deviation σ. If this step is below a threshold it is ignored, else, it is summed up with the present value. Then, if the resulting value is below 0 or above 1, it is rescaled with a remainder operator so as to stay circular (as an

example, -0.2 equals 0.8). In the self-adaptive version, for each fitness type, a separate standard deviation value is stored for each DU. These values are mutated with a creep mutation operator before the nudge mutations are carried out

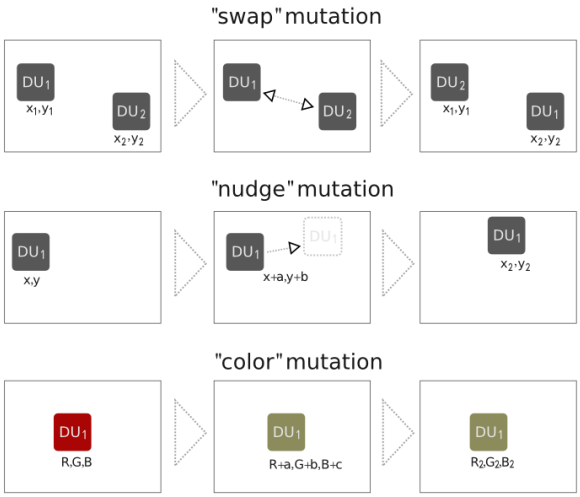


Figure 4.4 : Mutation operators.

For the application series, five procedures have been implemented for fitness calculations. Two of these functions are combined together with a weighted sum. Hence, there are four fitness types:

1. Out + Overlap fitness: For the applications, a flexible spatial representation is used, which permits the DUs to overlap each other or to cross the outer boundary of the canvas. This objective function is implemented as a combined function, to control the outward migration and overlapping of the DUs. Fitness value for this objective is calculated as follows (Figure 4.5): Each DU has a rectangular bounding box. For each candidate layout, first, all DUs are checked for whether their bounding boxes violate canvas boundary. If a DU is partially violating the boundary, a penalty is given according to the ratio of the violating part. If a DU is totally out of the boundaries, a penalty is given according to the distance from the center of the image. Secondly, all DUs are checked for overlapping other DUs. If overlap occurs, the overlapping area is divided by the smaller DU's area, and a penalty is given accordingly. Finally, total outer boundary penalties and DU overlap penalties

are combined into a single fitness value by a weighted sum. In the implementations, nudge mutation is used along with this fitness type.

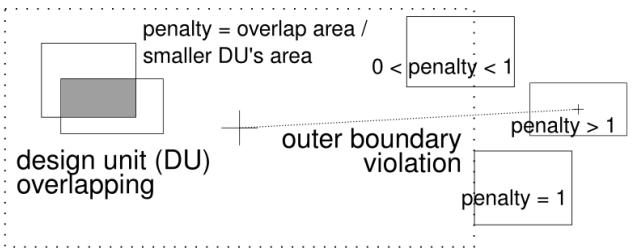


Figure 4.5 : Penalty calculations for Out + Overlap objective.

2. Cell fitness: First, a target feature has to be extracted from a target image layout. Values of the center coordinates (x, y) of each DU within the target image are divided by a pre-specified cell-width value, and a symbol representing the DU's type is placed in a grid-cell-dictionary. This dictionary is stored as the fitness target (Figure 4.6). For evaluation, the same procedure is applied to each candidate layout, and grid-cell-dictionaries for each of the candidates are obtained. Then, the two cell dictionaries are compared. For each cell, a penalty is given for the mismatches. If a DU type is present in a specific cell of the target, but not in the corresponding cell of the candidate, a penalty is applied. This fitness is meant to be tolerant for abundance cases, where candidate images contain larger numbers of DUs than the targets. Swap and nudge mutations are preferred for this fitness.

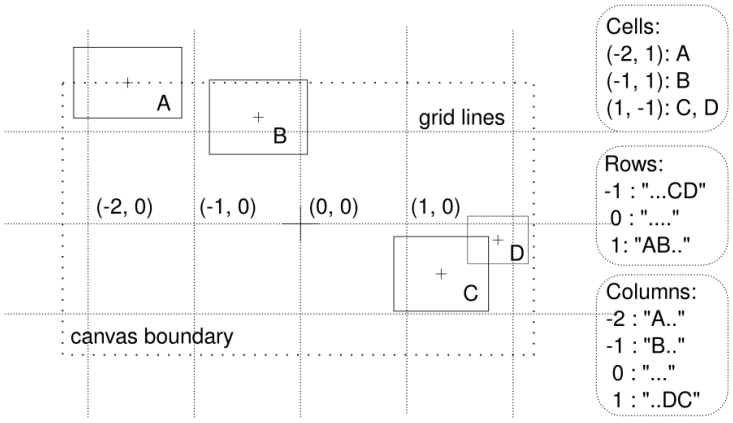


Figure 4.6 : Cell, row, and column preparation for Cell and Sequence objectives.

3. Sequence fitness: A grid is prepared for a specified width, then for each row and column, symbol strings are produced according to sequences of DU

types. Dots ('.') are placed for empty cells. This procedure is applied to both targets and candidates (Figure 4.6). For fitness evaluation, each of a candidate image's strings is compared with the corresponding string of the target. This comparison is done by finding the Levenshtein distance, which gives the minimum number of editing operations needed to transform one string into the other. The permitted operations are insertion, deletion, or substitution. This fitness has the advantage of taking relative positioning of the DUs into account, but it penalizes abundance cases. As with Cell fitness, swap and nudge mutations are preferred for this fitness

4. Color fitness: For this fitness type, a bitmap image is specified by the user as a target (Figure 4.7). For target preparation, first the RGB values of the target image are converted into HSV mode, which is relatively more intuitive for human perception. Then, for each series of H, S, and V values, separate histograms are prepared and normalized, so that the total area of the histogram equals 1. This step is necessary to compare images with different pixel quantities. For the histograms, several bin widths were tested and no considerable difference was observed. In the end, histograms with 70 bins are used. After the same procedure is applied to prepare candidate histograms, Manhattan distance is calculated between target and candidate histograms. Then the three distance values corresponding to H, S, and V histograms are combined with a weighted sum (equal weights are used). Color mutation is used together with an optional nudge mutation for this fitness.

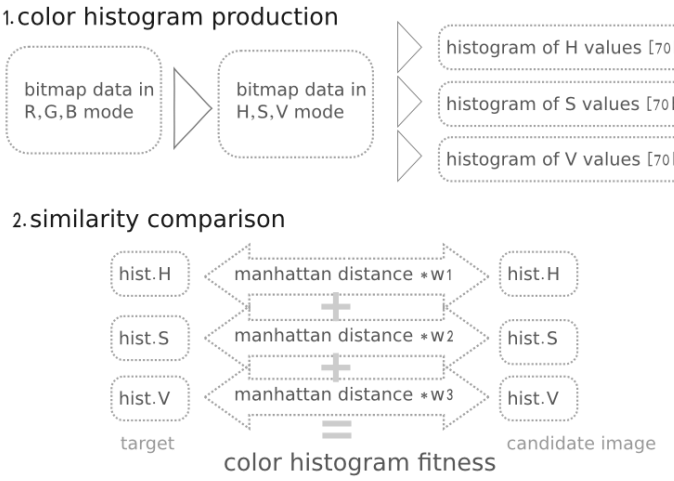


Figure 4.7 : Color fitness calculations.

There are a series of process parameters, which should be initiated before evolution starts[19]. Care should be taken for the adjustment of these parameters, because success of the process is closely connected with proper settings (Figure 4.8).

[19] Interleaved EA and d_p are implemented with the Python language. In addition to standard Python libraries, a set of external libraries are utilized, i.e., Numpy and Scipy for scientific computing, Matplotlib for plotting the process graphics, PIL for raster image processing, Pycairo for the generation of new images, Polygon and Shapely for geometrical calculations, and finally Psyco for speeding up the program.

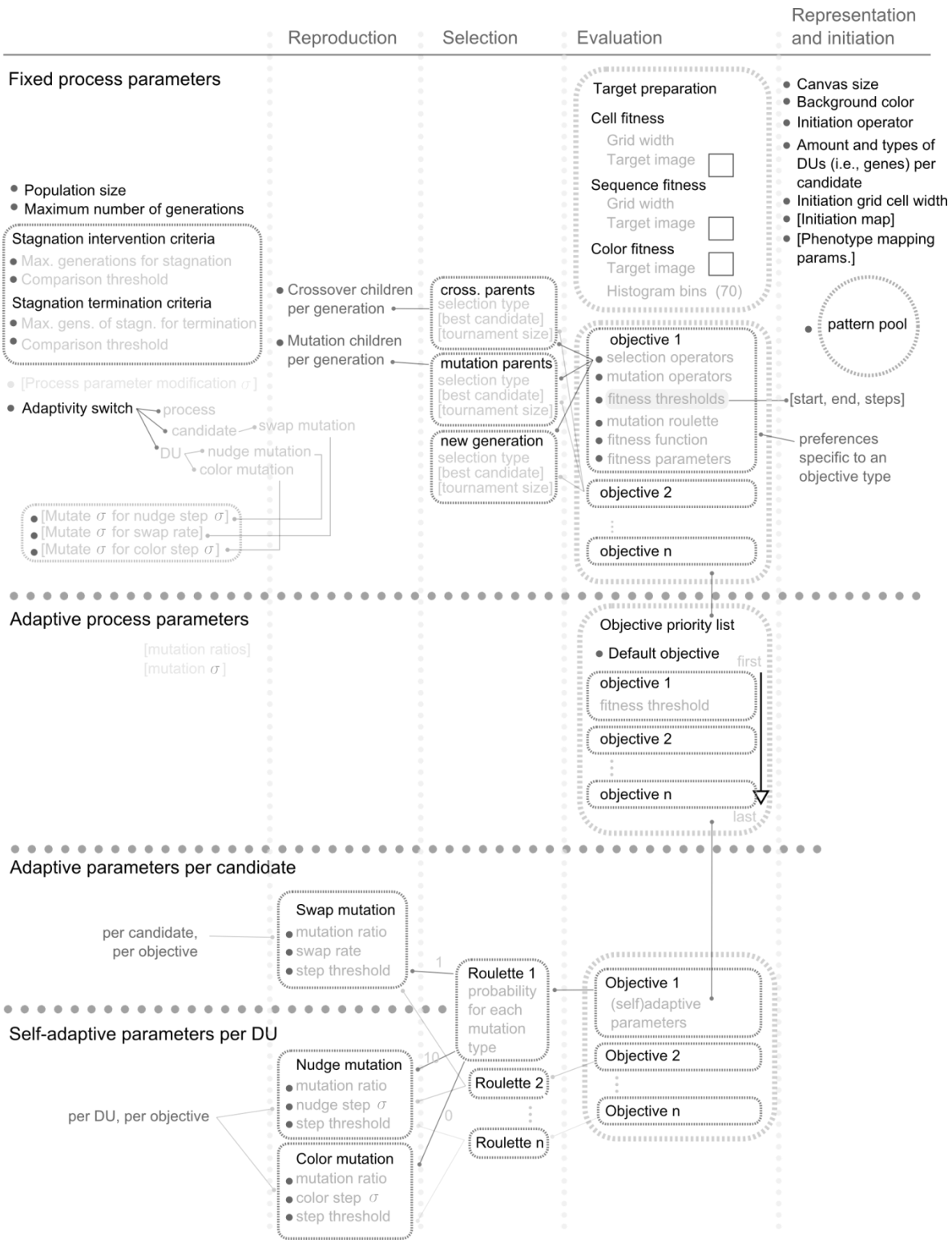


Figure 4.8 : Basic parameters of the d_p.graphics application.

As can be seen in Figure 4.8, there are fixed and adaptive parameters. Adaptive parameters are in turn categorized under three headings, i.e., parameters per process, candidate, and DU. Fixed parameters are determined before the process starts. For the adaptive parameters, initial values undergo changes throughout the evolutionary

process. Basic process parameters, such as population size, are fixed. Generation (tour) count determines the maximum number of generations before the process terminates. There is a second termination mechanism, i.e., the stagnation criterion, which will be described below.

There are switches that determine which elements will be taken as adaptive, non-adaptive, and self-adaptive. This gives fine-grained control over different aspects of the process. Another option controls the use of an initiation map. For the initiation of the candidates, pattern stamps have to be chosen beforehand, along with a basic DU list enumerating the types, numbers, and basic parameters regarding DUs.

An important adjustment concerns the priority list for the objectives. This list determines the initial preference for the objectives, in other words, which objective's process will be run first. In the d_p.graphics application, for each objective, there are two fixed fitness thresholds for start and end values. A third threshold, which is adaptive, starts from the start value, and throughout the process ascends towards the end value in equal increments (usually 5 or 10 steps). This parameter checks for the average fitness level of the population regarding an objective. Its usage will be further detailed below. In the d_p.layout application, the adaptive threshold approach has been discarded in favor of a random shuffling of the priority list in cases of extended stagnation. In either version, the priority list undergoes changes adaptively during the process. Together with this list, an additional dictionary holds parameters for each objective. These include parameters for the fitness function, mutation roulette (determines the mutation types and their relative probabilities for each objective), and selection operators for each selection stage. All selection operators offer an option to choose a number of best candidates directly. Additionally, there is a tournament size parameter for tournament selection.

A separate number can be assigned to each objective for the offspring to be produced through mutation. There is no inherent limit on this number; it may exceed the population size, which enables the examination of a wider search space with the expense of computing time. A higher value is usually useful; however, it might also be desirable to diminish this value, especially for the more time consuming objectives.

A series of parameters are dedicated to stagnation controls, and are used for two main aims. The first aim concerns the termination of a completely stagnated process. If a population within an EA stops improving its average fitness levels, this usually means that either the EA is not working properly, or it has reached a level after which a considerable fitness improvement is not expected. To check for such situations, a stability threshold is determined and at each generation, a stagnation control is applied to each objective. In d_p.graphics, the improvement of a generation is determined over the improvement of its average fitness with regard to the previous generation. Improvement information of a pre-specified number ('stability stop') of generations is stored within a dynamic dictionary. Each generation is compared with the previous generation to check whether it improved or not. The ratio of unimproved generations to the stability stop is compared with a stability threshold. If it is below the threshold, the objective is assumed to have stagnated. If all the objectives have stagnated, the process terminates.

There is another stagnation control mechanism only for the objective that is determined as the current objective. Several alternatives have been tested for this mechanism. In the adaptive version, there are two stages. In the first stage, if the population is found stagnated with regard to the current objective for a pre-specified number of generations (usually in none of the last 5 generations), then adaptive process parameters are modified. However if the stagnation is found to have exceeded a second threshold (usually double the first number, i.e., 10), then the objective priority list and objective thresholds are modified using a random number drawn from a Gaussian distribution with a pre-specified standard deviation (process parameter standard deviation). In the self-adaptive version, if the population is found stagnated with regard to the current objective for a pre-specified number of generations, the process directly goes on to modify the priority list. The modification procedure will be described below. The process for the Interleaved EA can be observed in Figure 4.9. It should be noted that, a different stagnation test has been implemented for the d_p.layout application, which is measured over the slope of the fitness graphs. In this case, the stagnation threshold takes the form of a minimum slope value.

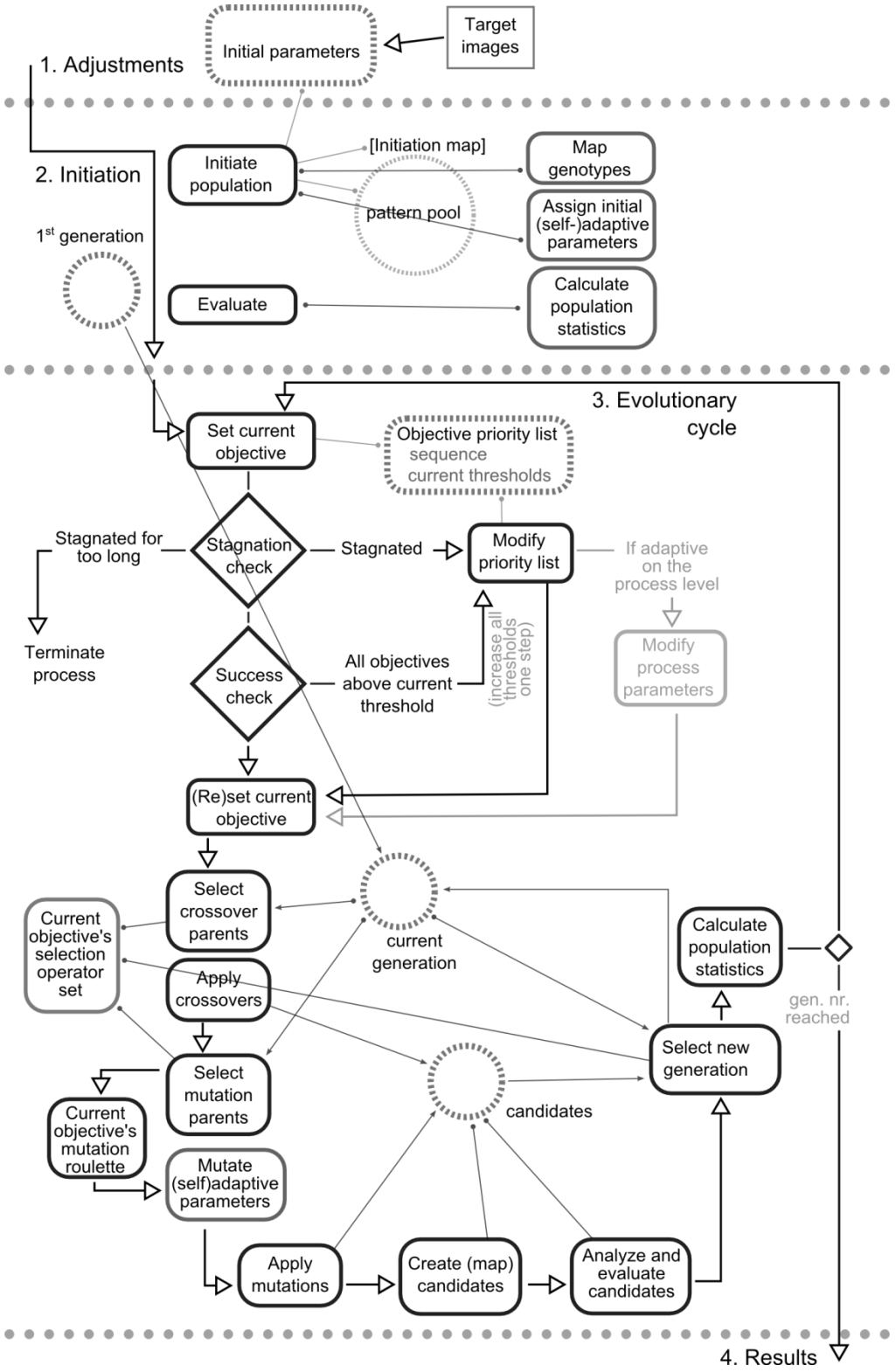


Figure 4.9 : Evolutionary process for the self-adaptive Interleaved EA for the d_p.graphics application.

Additional information has to be provided by the user before the evolution starts; most importantly, fitness measurement mechanisms and basic parameters for each objective.

For the Out + Overlap objective, which controls the amount of DU overlapping, there are two separate penalty parameters for overlap and boundary violation conditions. These parameters should be carefully weighed with regard to each other.

For the Sequence objective, first, a target image has to be supplied and processed as was described above. Through the processing, two dictionaries are prepared such as:

- target_rows = {0: '.SS...MSS', 1: '.SSB..MSS', -2: 'SSSSSSSSMSSSS', -1: '.S...MS', -3: '.S....S'}
- target_columns = {0: '...SS.', 1: 'MMMSM.', 2: 'SSSSSSSSS', -2: 'B..SS.', -4: '...S.', -3: 'SSSSSSSS', -1: '...S.'}

The numbers specify the numbers of rows and columns. Each letter in these dictionaries concerns a type of DU. In the current application, types of DU's comprise small, medium, and large pattern stamps along with text stamps. For the parsing of the candidate layouts, a grid-width is specified. There is also an option to give different weights to row and column ratings.

Adjustments for the Cell objective are similar to the Sequence objective. This time a dictionary of cells is produced from a target. The target can be the same as the Sequence target.

For the Color objective, first a target image is processed to obtain a target histogram. Relative weights may be specified for the values of hue, saturation, and value, which can help favor one over the others. An important parameter is the downscaling rate, which scales down a candidate before evaluation in order to save time. After several trials, a rate of 0.4 is fixed for applications. The number of histogram bins is equaled to the number of bins used for the preparation of the target images. Finally, for the Pareto-based version, there is an option to scale the effect of Pareto ranking up or down.

The first generation of candidates can be initiated either randomly or by using an initiation map as a probabilistic guide that locates stamps within a grid. Color values are assigned randomly. During initiation, values for candidate based adaptive parameters are initiated using the initial values. Likewise, the DU based self-adaptive parameters are initiated using the initial settings. It should be noted that for each of these types of parameters there are separate parameters for each of the objectives. In the current application, candidate-based parameters include the swap rate while the

DU based parameters are nudge step and color mutate step. After initiation and mapping, the individuals within the first generation are evaluated according to all objective types listed in the objectives list. The results are stored in the genotypes of the candidates. Additionally, overall process statistics are calculated and stored in log files.

As the first step of evolutionary cycles, the default objective is set as the current objective. There is a success check at this stage. As described above, there is an average fitness threshold for each objective. Each objective is examined to find whether it surpasses this threshold. If the average fitness values of all the objectives are found to surpass their thresholds, all thresholds are increased one step. The reason for this operation is related to the determination of the current objective. For the determination of the current objective, the priority list is traversed to find the first objective where the mean fitness is below its current threshold. The first objective that is found to stay below its upper threshold is determined as the current objective (Figure 4.10). In this way, when the current fitness surpasses its threshold, the next fitness type is activated, and when, because of possible conflicts between the objectives, the current objective's operations decrease the mean fitness value of a prior objective, the prior one is re-activated to readjust the population.

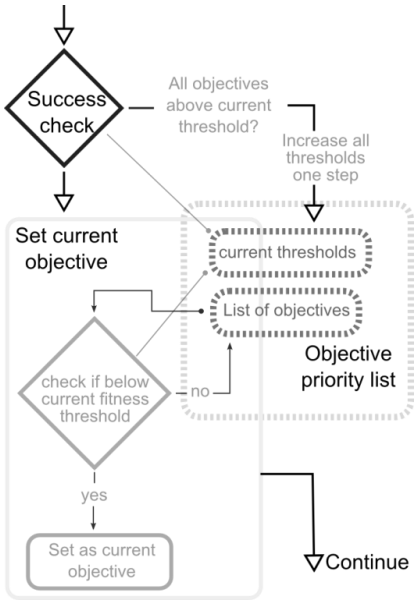


Figure 4.10 : Adaptive determination of the current objective.

The aim of this mechanism is to gradually improve all objectives simultaneously. If the prior objectives have already achieved their temporary aims, then the process

switches to the next objective. This mechanism for using feedback from the process was intended to create a dynamic decomposition scheme peculiar to each design task.

After the default objective is determined, stagnation controls are carried out. During evolution, if the process is found to have stagnated for a pre-specified number of generations, adaptive process parameters are modified. For the self-adaptive version, several modifications are carried out over the priority list. When stagnation occurs, if the stagnated objective is not the last one on the list, its position is exchanged with the next one on the list; thus, the priority list is dynamic. This way, another objective may be assigned as the current objective, if it is below its threshold (Figure 4.11).

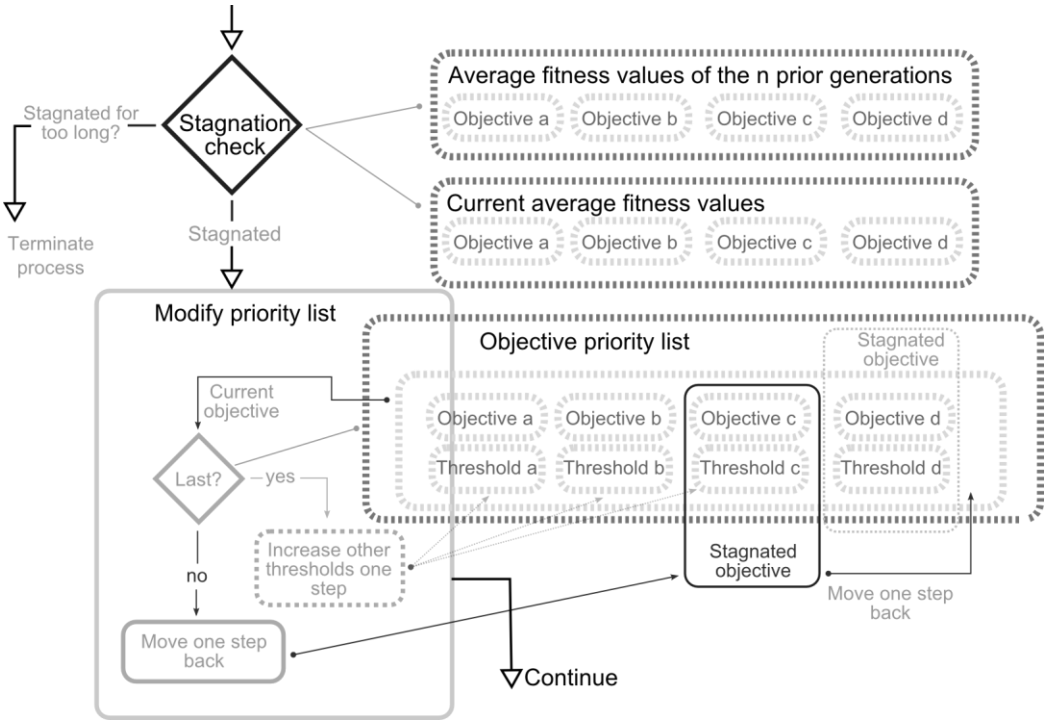


Figure 4.11 : Modifications in case of stagnation (for d_p.graphics implementation).

In case of stagnation, if the current (stagnated) objective is already at the end of the priority list, this means the prior objectives have achieved their intermediate goals, but the last objective stagnated. To steer the population towards other areas of the search space, the thresholds of the other objectives are increased for one step. This procedure will be repeated if stagnation continues, until a prior objective is set as the current objective.

However, due to the specifics of each design task, the success and speed of the objectives vary considerably. Consequently, a process may start with the first objective, then move on to the second and come back to the first repeatedly, never

getting to the third or fourth objective. Therefore, in this approach initial settings for priority list and fitness thresholds become critical for the success of the algorithm. However, for each new problem the fitness topography considerably changes, making a prior inference of the threshold values impossible. This necessitates extensive testing for each new problem, which is by no means desirable. Indeed this would be disastrous in real world applications. This is why, in the d_p.layout application, the adaptive threshold approach has been discarded in favor of a random shuffling of the priority list in cases of extended stagnation, which, together with the rank-based selection procedure, totally dispenses with the prior adjustment requirement.

After these procedures, the candidates for crossover and mutations are selected using the current objective's selection operators and parameters. Then the recombination operations are carried out. For the 'n point crossover' operator, the number of DUs to be exchanged between two parents is an important parameter, which can be evolved as an adaptive process parameter.

For the adaptive and self-adaptive versions, the mutations of the adaptive parameters are carried out just before the mutations of the DUs. Each objective has a separate roulette for guiding the mutation operations. The roulette determines which mutation operators will be used for that objective and in which order. It also determines the selection probability of each mutation operator for that specific application. More than one mutation operator can be applied to a candidate sequentially, which is also controlled by the roulette of the objective.

After the new genotypes are obtained, they are mapped to their phenotypes, and evaluated before they are added to a secondary pool of candidates. At this stage, there are two pools of candidate images; first, the current generation, and second, the offspring. The population size is kept static; therefore, a selection procedure is applied to obtain a single population, according to the selection operator specified by the current objective. The process iterates through these steps until a predefined count is reached or the population stagnates for a pre-specified number of generations for all objectives.

The implemented Pareto-based approach is rather simple. It is based on Fonseca and Fleming's proposal as described by Back, Fogel, and Michalewicz (2000).

Candidates are simply assigned a cost value according to the quantity of other candidates in the population that dominate them. Fitness of a candidate is determined by negative cost value. No strategy is implemented for guiding the search towards any sub-region or for an even distribution of the Pareto front.

4.1.5 Application cases for d_p.graphics and the naive Interleaved EA

In this section, the initial tests and demonstrations for the d_p.graphics and the naive version of the Interleaved EA will be presented. The tests will be carried out through the applications for three distinct design scenarios, i.e., the generation of decorative patterns for computer desktop wallpapers, coffee cups, and T-shirts. The aim of these tests is to verify that the evolutionary operators of d_p.graphics carry out their tasks as expected and naive Interleaved EA works reliably for a variety of scenarios. The series of tests will concern the following questions:

1. Do the fitness procedures guide a population of solution candidates towards a desired state?
2. Do the fitness procedures work as expected when they are used together in multi-objective evolution?
3. Does the naive version of Interleaved EA operate as expected? What are the evolutionary characteristics of this system?
4. With respect to parameter control, are the process-based and self-adaptive schemes advantageous over the regular case?
5. Is the approach flexible in its totality? Is it easily adaptable to different scenarios and cases?
6. How does the approach compare to a Pareto-based approach?

For the first three questions, single-objective and multi-objective test series have been carried out through several design scenarios. Results of the tests will be examined through (1) average fitness graphs for each objective, which display the progression characteristics of EAs, and (2) visual evaluation over the results, which better indicate whether expected results are obtained or not.

For the fourth question, process-based and self-adaptive versions have been compared with the non-adaptive (regular) case. The adaptivity scenario is meaningful

for cases where, (1) best parameter combinations are not known, and (2) where the best parameter combinations change during an evolutionary process. Therefore, a series of tests will be carried out with both poor and 'best-so-far' parameter settings. The former case mainly tests adaptive parameter-control functionality, while the latter indicates whether parameter adaptivity is also advantageous throughout the process.

Three different scenarios have been devised for answering the fifth question. There are two sub-tasks for each scenario. Additionally, for each of the six resulting scenarios, both 'abundant' and regular versions have been tested. The notion of abundance here refers to the number of DUs used for each task and is employed to examine the flexibility of the layout arrangement functionality.

Finally, for the sixth question, a simple Pareto-based version has been implemented, in order to compare its results with former trials.

Some of the above-mentioned tests have to be carried out in combination. For example, each of the three options for adaptivity (regular, process-based, and self-adaptive) has to be tested for single and multi-objective cases as well as for regular and abundant cases. These in turn have to be compared with the Pareto-based case. This makes the real testing and demonstration process slightly more complicated. Nevertheless, all six aspects will be addressed in the following pages, though in a combined and comparative manner.

4.1.6 Single-objective tests and adaptivity

Usually, a series of initial tests are carried out before the real tests, in order to find the best settings for important process parameters. Ideally, the parameter-tuning process should concern combinations of parameters, as these are not independent from each other. This requires a series of tests for each combination. However, as there are a large number of important parameters, the quantity of the required tests increases combinatorially. Therefore, in practice, most of the parameter finding process has to be carried out as if the parameters were independent. This considerably reduces the time spent for these tests, at the expense of systematicity.

In this manner, a best-so-far parameter combination has been obtained for each scenario through initial runs. With these best-so-far parameter settings, single objective evolutions have been carried out to assess the performance of each of the

fitness procedures separately. Three series of tests have been carried out with different population size and mutation count combinations. The first series concern a population size of 10. At each generation, 20 candidates are generated through mutation. The second and third series tested 40 – 40 and 80 – 80 combinations for population and mutation numbers.

For the layout objectives (Cell and Sequence), vector images have been produced as targets. For the Color objective, bitmap target images have been prepared. The candidates are initiated randomly, using the same number and types of DUs with the used layout target.

In each series, for each objective type, both non-adaptive and process-based adaptive versions have been tested. For each test type, 10 runs are carried out. Due to the stochastic nature of EC, this number should have been higher, ideally at around 100. However, image evaluation was time consuming and for this task setting, initial tests displayed a regular evolutionary progression; this means, fitness graphs were showing that the courses of the evolutionary runs were mostly stable. Considering the large number of required experimentations, each of the test series was limited to 5 or 10 runs, which appears sufficient in this case, at least to illustrate the general course and capabilities of the approach (In the d_p.layout application, where a real world application is aimed at, such tests are carried out over 100 runs). For the single objective test series, each run lasted for 200 generations. Initially, 60 tests have been carried out (2 options for process-adaptive and non-adaptive x 3 options for population size and mutation count combinations x 10 runs). All runs have been initiated with the same initiation values.

Figure 4.12 shows target images, example results, and average fitness graphics for each objective function, for the first case (population size: 10, mutation count: 20). Example results are given on the leftmost column. Each evolutionary run yields a series of such images. Each image is an example result of a single-objective run. The two columns on the right include average fitness graphs for non-adaptive and process based adaptive evolutionary runs. Each graph is obtained by averaging out 10 runs. In the graphs, the vertical dimension shows fitness levels, while the horizontal dimension indicates the generation count. In addition to the progression of the average fitness levels, best and worst candidates are given to indicate population distribution.

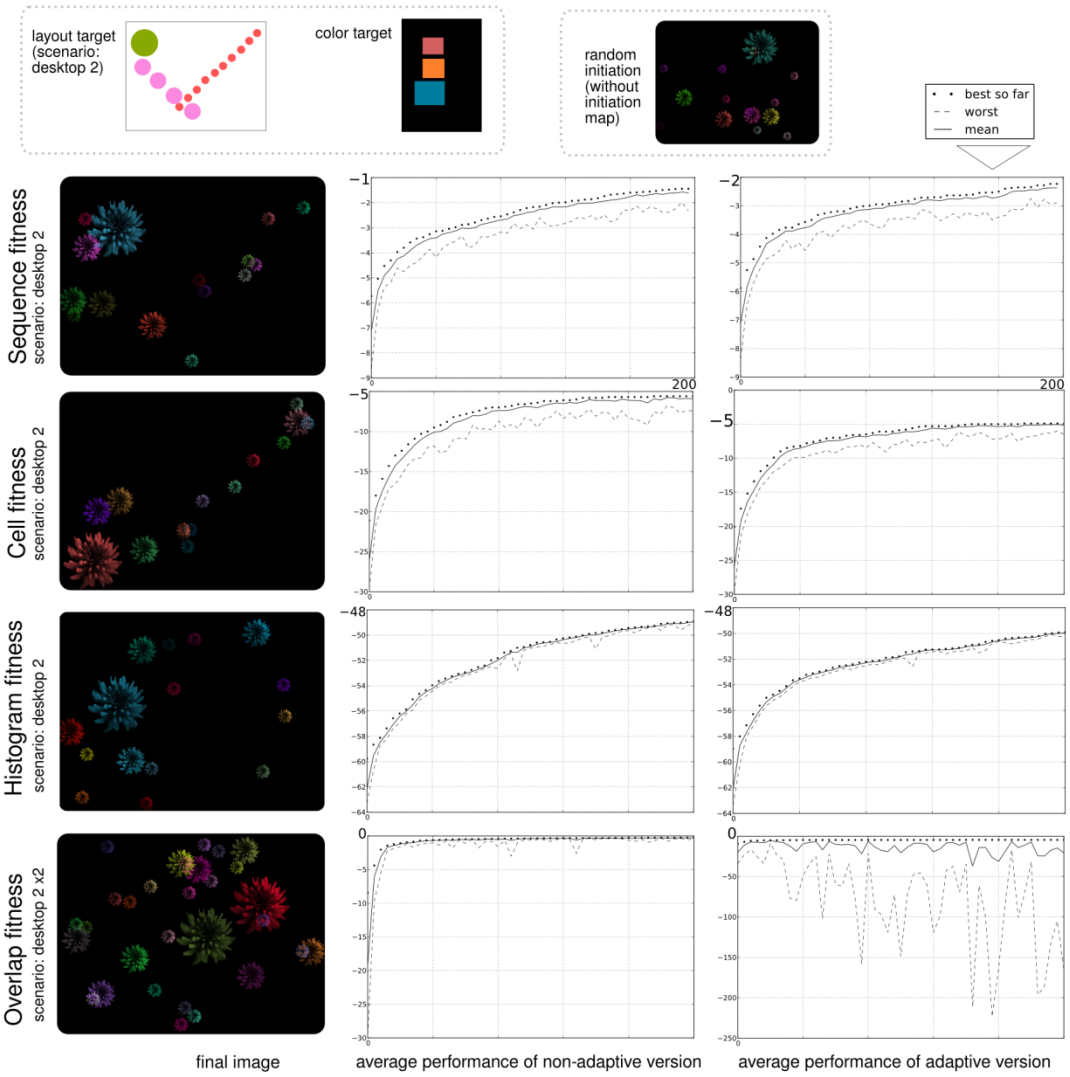


Figure 4.12 : Single objective test series through the desktop wallpaper scenario; 10 runs of 200 generations for each objective. Non-adaptive and process-adaptive versions are given, both use the best-so-far initial parameters.

It can be seen by a visual examination of the examples that the graphic layout objectives (measured by Cell and Sequence fitnesses) managed to evolve the randomly initiated population towards the desired compositional state. Cell objective appears to work better but converges more quickly. Sequence objective on the other hand, although slower, was continuing its fitness improvement. The controlling Out + Overlap objective appears to successfully keep the DUs within boundaries and distribute them in order to minimize overlaps. The visual success of the Color objective is not well understood in this scenario, it can be better examined in the subsequent trials.

A fine slope of fitness progression can be observed in most of the fitness graphs, with the exception of Out + Overlap objective, which converges too fast, partly

because of the simplicity of the problem. A high and continuous slope indicates that better results would be obtained for longer generation counts.

In general, with the best-so-far parameters, the process-adaptive versions appear worse for Sequence and Color objectives and only slightly better for Cell objective. With the exception of Out + Overlap objective, all the objective functions work reliably with both adaptive and non-adaptive versions. For this test series, the process-adaptive version did not work reliably for the Out + Overlap objective. As can be observed in the bottom right graph in Figure 4.12, the average fitness level fluctuates instead of improving and the increasing distance between the fitness levels of the worst and best candidates indicates that the population is wildly dispersed within the search space.

In the second case (population number: 40 – mutation number: 40), again non-adaptive versions worked (mostly) slightly better, and in the last case (population: 80 – mutation: 80), performances of adaptive and non-adaptive versions became roughly equal. However, the increase in population, while lengthening the evolutionary process, did not result in considerably better fitness levels. This was one of the reasons for limiting the population sizes for the following experimentations.

The process-based adaptive version modifies the process parameters whenever the process stagnates; however, as was discussed in the previous sections, the modification process itself is not guided. Additional knowledge is required to decide whether the parameter should be increased or decreased. In the non-adaptive and self-adaptive versions, there are still adaptive parameters, such as the settings for the objective priority list, which is an essential component of the naive Interleaved EA. However, the mutation parameters are either not evolved or are evolved through self-adaptivity, i.e., as attached to candidates or DUs.

In order to compare the relative performances of the non-adaptive and self-adaptive variants, series of tests have been carried out with single-objective runs, using the same initial parameters; again, 10 runs for each configuration. In Figure 4.13, several series of tests with various population and mutation count combinations are presented. The graphs concern single objective runs for the Cell objective. The initial parameters are tuned to the best-known parameters. With these settings, performances of the non-adaptive and self-adaptive versions do not exhibit a

consistent pattern of differentiation. When the best initial parameters are used, an increase in the population and mutation counts did not result in an improvement in the attained fitness levels.

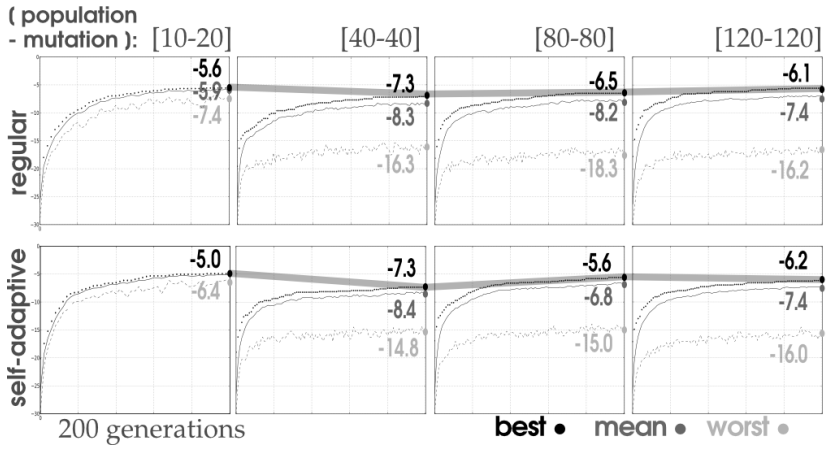


Figure 4.13 : Comparison of non-adaptive and self-adaptive versions with best parameter settings. Single-objective runs for Cell objective. Each graph is the average of 10 runs.

The expediency of adaptivity manifests itself only when ‘poor’ initial parameters are used (Figure 4.14). A parameter configuration is poor in the sense that it is tested at least 10 times for an objective and resulted in considerably lower fitness levels than attained best levels. With poor parameters, self-adaptive version performs significantly better with a negligible additional performance cost (Figure 4.14). Additionally, these tests show that, contrary to the best parameter settings, for the poor parameters, an increase in the population size has a positive effect on EA performance. Yet, even for small population numbers, the self-adaptive version still performs almost as good as the regular version with the best parameters (Figure 4.14).

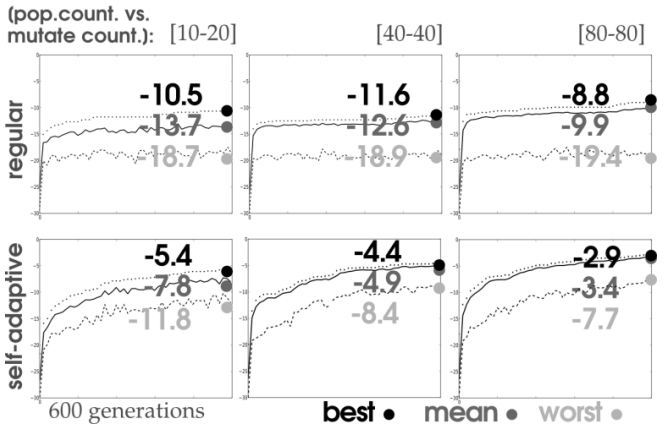


Figure 4.14 : Comparison of non-adaptive and self-adaptive versions with poor parameter settings. Single-objective runs for Cell objective. Each graph is the average of 10 runs.

4.1.7 Multi-objective tests for abundant and regular cases

As was shown, the objectives work as expected and the self-adaptive version is a viable method for parameter-control. The second stage of tests concerns multi-objective evolution for assessing the overall viability of naive Interleaved EA and d_p.graphics. For the multi-objective test series, twelve real-world scenarios have been developed. These concern the generation of decorative arrangements for coffee-cups, T-shirts and computer desktop wallpapers. For each scenario, target images have been produced for color and layout objectives.

Six of the scenarios use exactly the same amount and types of DUs with their layout targets, while the remaining scenarios involve exactly two times this amount. The former are called regular or non-abundant cases, while the latter are abundant cases. The abundant cases are generated in order to test the flexibility of the task definition (Figure 4.15).

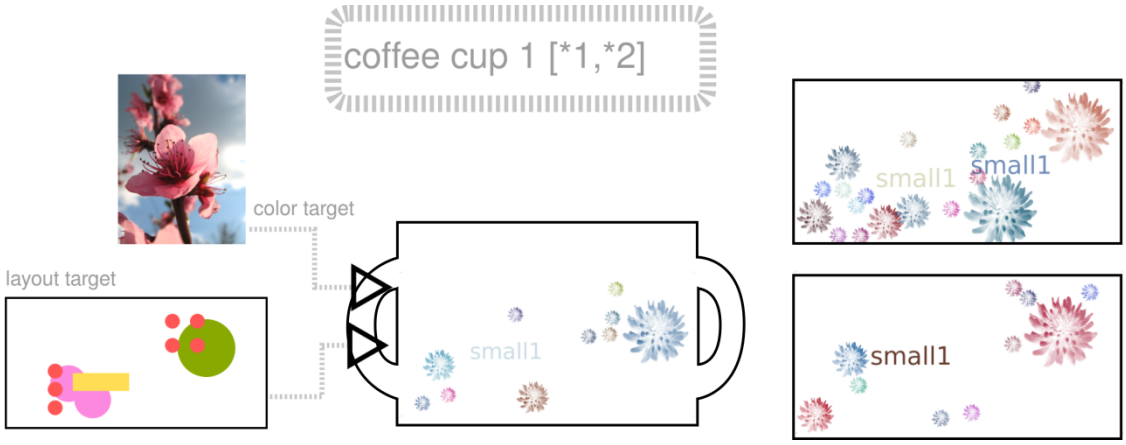


Figure 4.15 : Inputs and results of the first Coffee-cup scenario, regular and abundant (upper right) cases.

Initially, multi-objective test series have been carried out with the process-based adaptive version of the Interleaved EA; 10 runs for each scenario and for its abundant version.

It should be noted that, technically, the proper adaptation procedure is different for each objective, and it should be developed in parallel with the objective function. This is one of the weaknesses of the process-based adaptive approach, and is illustrated by the failure of the Out + Overlap objective in the process-adaptive version (see Figure 4.12). Therefore, in the multi-objective tests, adaptivity is disabled for this objective while the other three objective types have been used in an adaptive fashion.

Satisfactory results have been obtained through the experimentations, which is reflected by the attained fitness levels. Figure 4.16 presents example results and process graphics for the second Coffee-cup scenario. It can be observed through visual evaluation that the resulting images reflect both of the target images and also the formal constraints implemented through the Out + Overlap objective. Results of the other scenarios are presented in Figures 4.17, 4.18, 4.19, and 4.20.

Layout objectives converged rather fast, as a result of initiation through initiation maps. After this fast convergence, while the Color objective continued to work for improving the color distribution, layout objectives simply maintained their fitness levels, which is exactly the aim of the naive Interleaved EA. It should be noted that the Out + Overlap objective was set to rather tolerant overlap values for these scenarios, which required a degree of overlap. These trials show that the stochastic nature of the EAs, when coupled with tolerant objective functions, exhibit a flexible

design approach. In other words, it is possible to produce a variety of similar, yet distinct solutions for a graphic design task; solutions, which comprise different types and numbers of DUs, while still resembling the same targets.

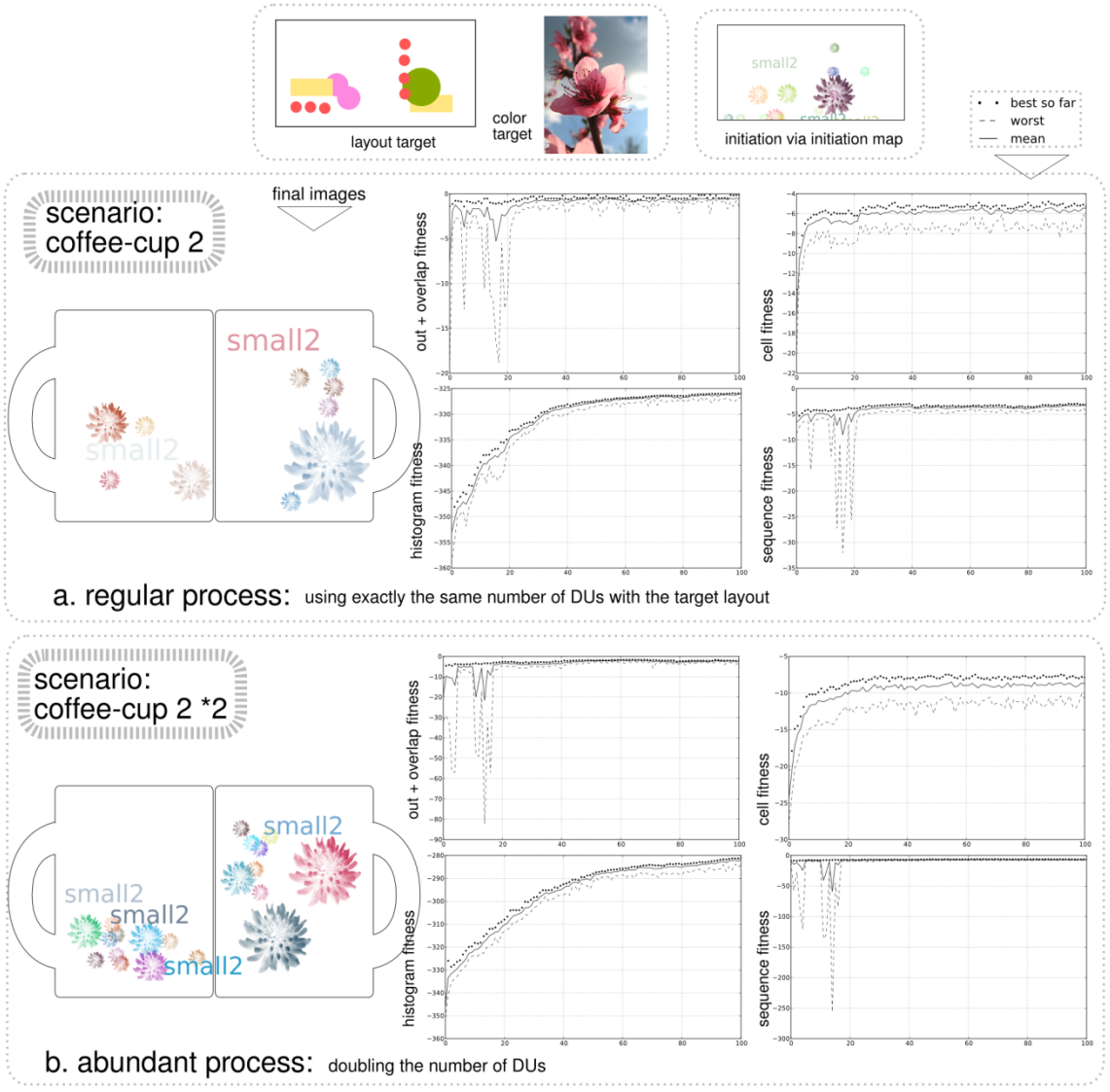


Figure 4.16 : Examples of multi-objective runs: Coffee-cup 2 scenario and its abundant version (process-adaptive).

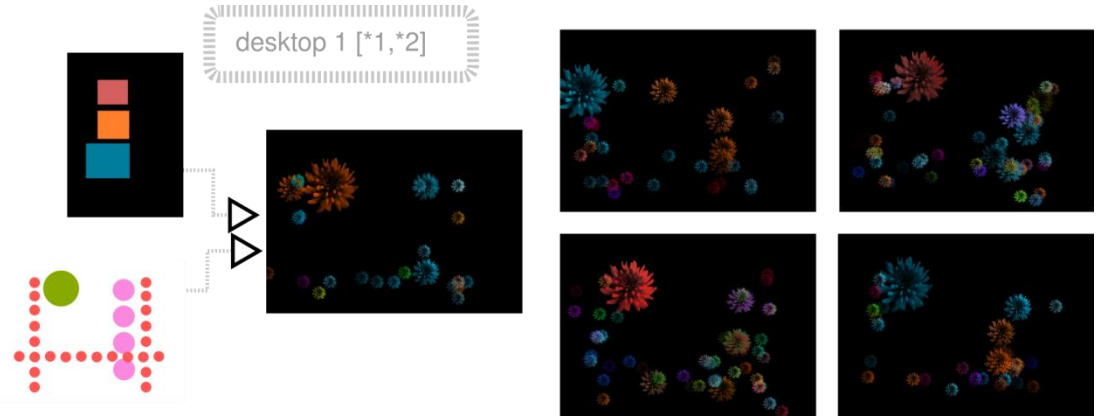


Figure 4.17 : Results of the Desktop 1 scenario, non-abundant versions (process-adaptive).

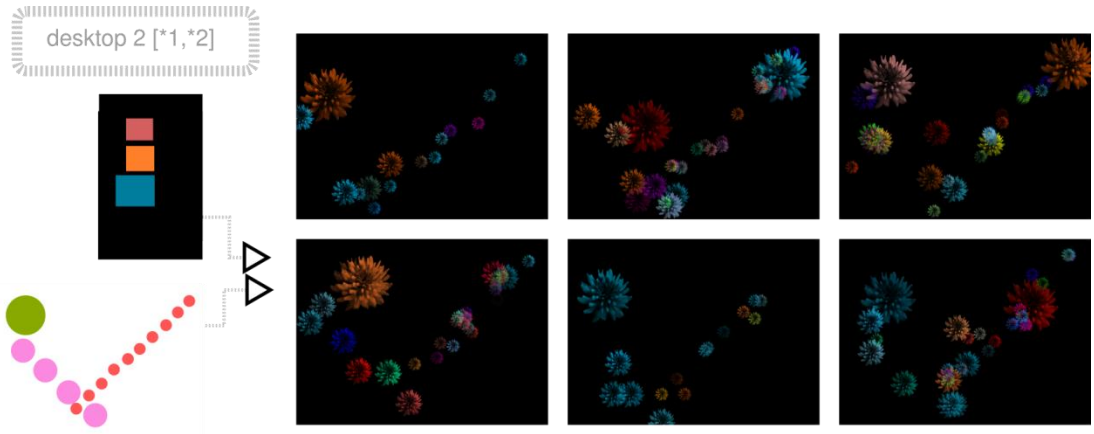


Figure 4.18 : Results of the Desktop 2 scenario, abundant and non-abundant versions together (process-adaptive).

Figure 4.19 : Results of the T-shirt 1 scenario, abundant and non-abundant versions together (process-adaptive).

Figure 4.20 : Results of the T-shirt 2 scenario, abundant and non-abundant versions together (process-adaptive).

4.1.8 Naive Interleaved EA vs. Pareto-based version

After observing the performance of the self-adaptive version, further multi-objective experimentations have been carried out for non-adaptive and self-adaptive versions. These test series have been carried out through one of the scenarios, using all four objectives. In the self-adaptive trials, self-adaptivity is enabled for all objectives, including the Out + Overlap objective. Different configurations are compared in order to examine three basic issues in combination:

1. Multi-objective comparison of the non-adaptive and self-adaptive processes.
2. The utility of initiation maps.
3. Comparison of the self-adaptive naive Interleaved EA with the Pareto-based version.

Figure 4.21 compares the results of these experimentations. The essential reason for the implementation of the adaptive variants was to alleviate the burden of the parameter-finding process. The below experimentations demonstrate the viability of the approach with regard to this aim. Interestingly, the performance of the self-adaptive version with poor initial parameters is almost equal to the non-adaptive version with the best initial parameters. Actually, it is slightly better for most objectives (Figure 4.21). However, the low number of trials prevents us from asserting this superiority in a certain manner. It should be noted that such superiority has not been observed in the systematic trials of the d_p.layout application, which will be further discussed in the next chapter. Nevertheless, when the potential of emancipated time of the user is considered, together with the negligible additional computation cost, self-adaptive procedures deserve further experimentation.

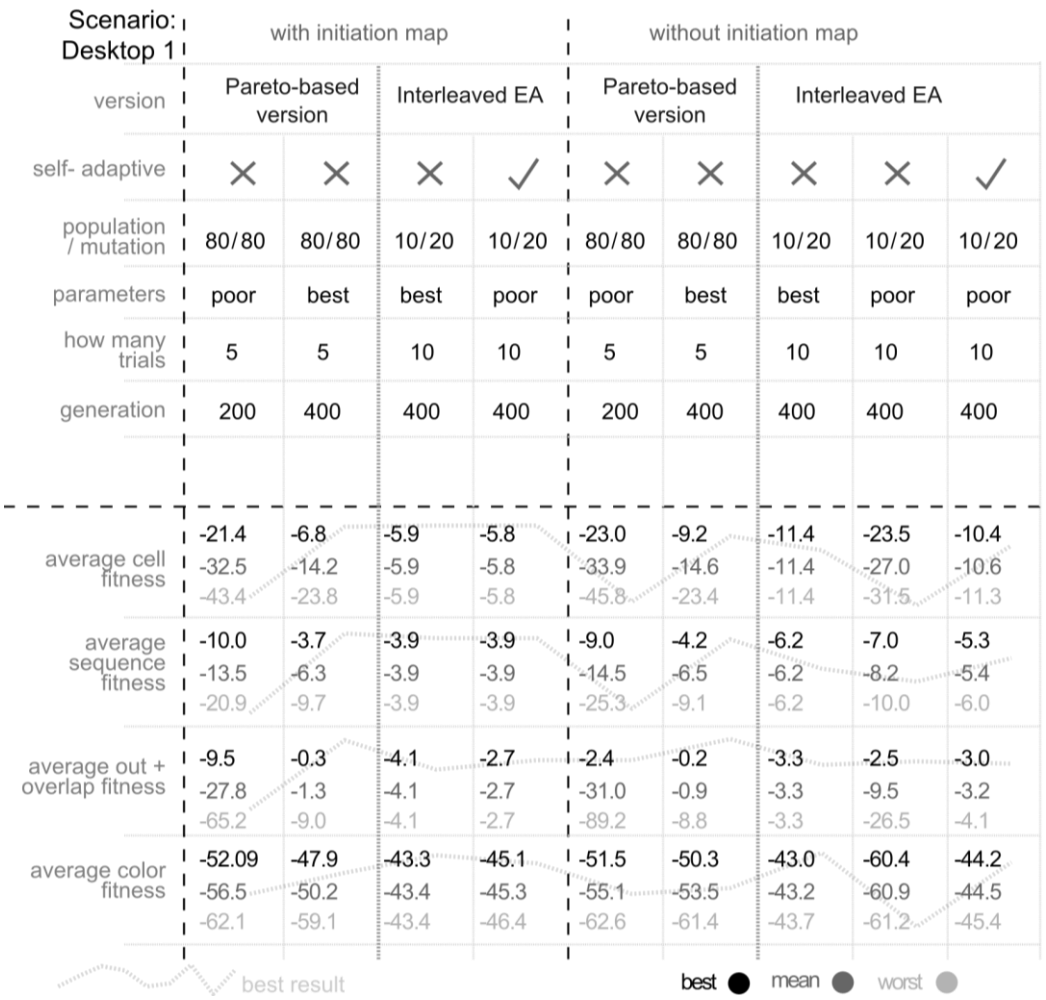


Figure 4.21 : Comparison chart for average performances of different version combinations.

The most important difference between the Pareto-based version and the naive Interleaved EA is that, in the Pareto-based version all objectives share the same mutation procedures and the same selection and initial parameter configurations. Contrarily, the Interleaved EA incorporates different mutation procedures and configurations for each of the objectives.

Another important difference is that, even with conflicting objectives, the Pareto-based version prefers a move towards a better area in the search space, and never worse. Yet, partly because our implementation lacks additional mechanisms, a simultaneous improvement for all the fitnesses is not guaranteed, and the populations tend to disband towards the fringes of the multi-dimensional search space. Each fringe includes better candidates in terms of a single objective. This result is contrary to what is desired in the design task, where the main objective is to obtain solutions, which are good on all objectives simultaneously. Consequently, for our tasks, an

increase in the mean fitness values of the populations is important rather than the best-so-far value of a specific objective.

A related problem is that, in general, when the population size is larger, variety increases. Thus, it becomes hard to find individuals that reach high levels on all the fitnesses simultaneously. The solution for the previous trials for the Interleaved EA was simple: the population size was kept small, so that the mean levels of the fitnesses are increased in parallel. However, this is not viable for more complicated problems, such as will be encountered in building layout evolution.

On the other hand, even for the simple graphic tasks, initial tests revealed that in order for the Pareto-based version to function properly, at least a population size of 80 was required; which seriously lengthens the process. When this population size is reached, the best results of each objective are almost as well as the results of the Interleaved EA that has a population number of 10 and a mutation count of 20 (Figure 4.21). However, these best candidates of the Pareto-based version are usually 'bad' in terms of the other three objectives. This problem can be examined in Figure 4.22, where four dissimilar images are obtained from the Pareto-based version, none of which is 'good' on all fitness levels. Contrarily, the typical Interleaved EA converges on a family of similar solutions that are good on all objectives. One reason for the failure of the Pareto-based version might be the simplicity of the current Pareto-based implementation. In the Pareto-based version, mechanisms for directing the search towards more desirable regions were lacking. Consequently, although the final fitness levels appeared high, in reality it was not possible to obtain desired images that are good on all objectives. In any case, the Pareto-based version required much longer computation time than the Interleaved EA versions.

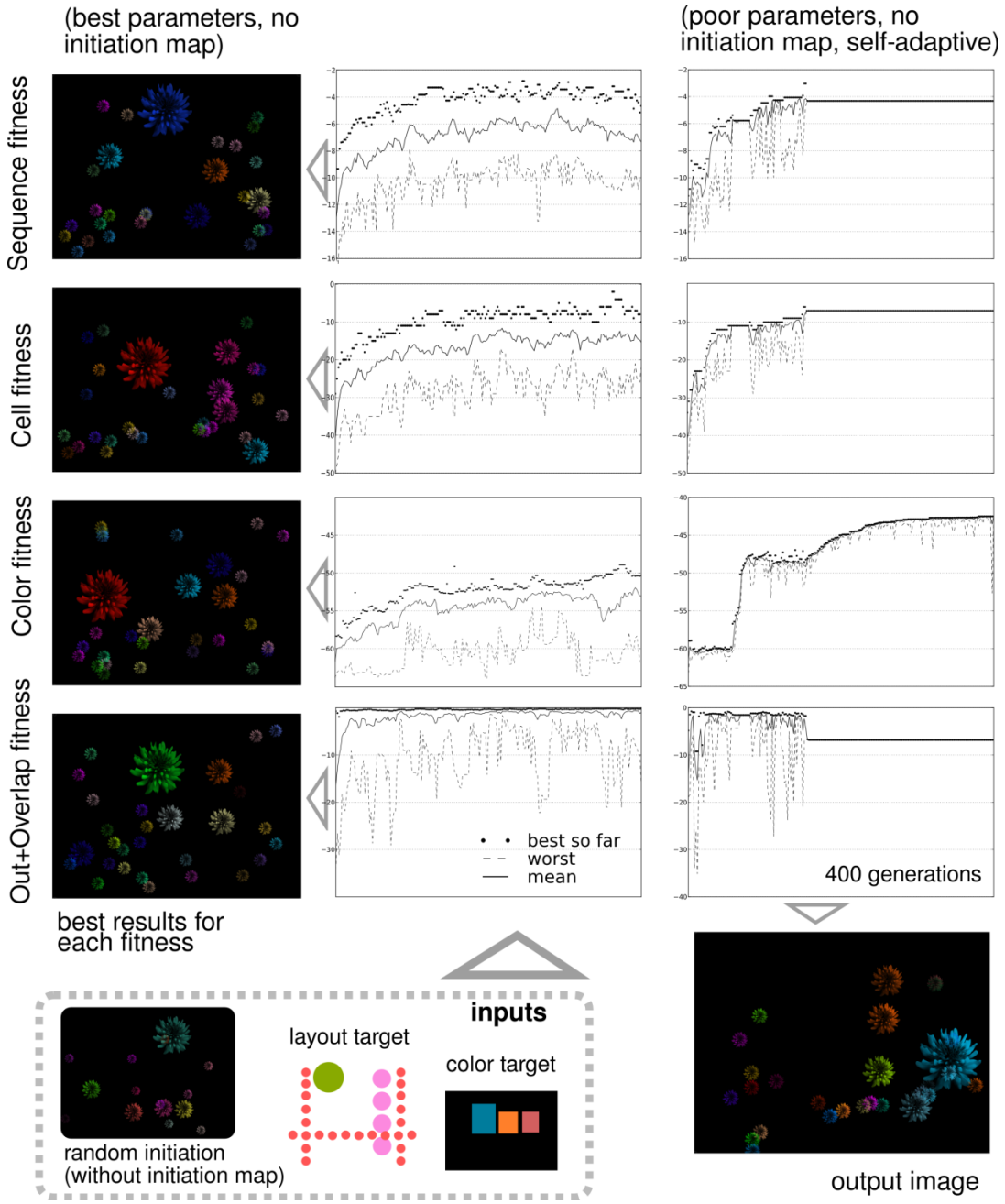


Figure 4.22 : Example processes and results for the Pareto-based version (left) and self-adaptive naive Interleaved EA (rightmost column).

The presented Interleaved EA approach is 'naive' in the sense that it ignores the interdependence of the objectives. To an extent, it behaves as if the conflicts or dependence between the objectives are not important. Nevertheless, for these task settings, the naive Interleaved EA approach works.

If the average and worst fitness levels are examined together with the best fitnesses, in Figure 4.21, it can be seen that the populations in the naive Interleaved EA tend to converge within a small range of fitness levels on all objectives simultaneously; exactly as desired. On the other hand, the populations in the Pareto-based version are

wildly dispersed. This can also be observed in the results that are exemplified in Figure 4.21. The typical courses and results of the two versions exhibit obvious differences. It should be noted that both versions would produce separate images for each objective, if they did not converge on a single individual.

It can be observed in the process graphs in Figure 4.22 that during the example run of the Interleaved EA, the three objectives have reached a high fitness before 200 generations, and the remaining period was spent for the Color objective. In the first half of the process, the downward spikes of the worst candidates indicate the phases where the current objective changed. In the Pareto-based version on the other hand, the range between the worst and best candidates is usually large.

As mentioned in the previous sections, for the naive Interleaved EA, two threshold values should be determined for each objective before each process. This has both advantages and disadvantages. As for the disadvantages, it might be difficult to adjust these priorities and thresholds. Several test runs may be required to find a reliable setting for each new scenario. For the graphic tasks, the initiation map mechanism alleviates this problem by starting on already high fitness levels for the layout objectives. An example run with an initiation map is presented in Figure 4.23. The map initiates the image layout, while the color values are initiated randomly. It can be observed that the Color fitness ("Histogram fitness" in Figure 4.23) constantly increases to reach from a random color distribution state to a higher fitness level, while the remaining objectives, which concern layout features, mostly work for preserving initial levels after an abrupt initial rise. The Out + Overlap objective has an additional task to stir up the initiated layouts, especially in the beginning of the evolution. Therefore, usage of the initiation maps alleviates the burden of threshold setting. Furthermore, initiation by maps generally leads to higher fitness levels as can be observed in Figure 4.21. The Pareto-based version is relatively less susceptible to the lack of initiation maps. This is partly because the fitness improvement in the first phases of the evolutionary processes is typically steep.

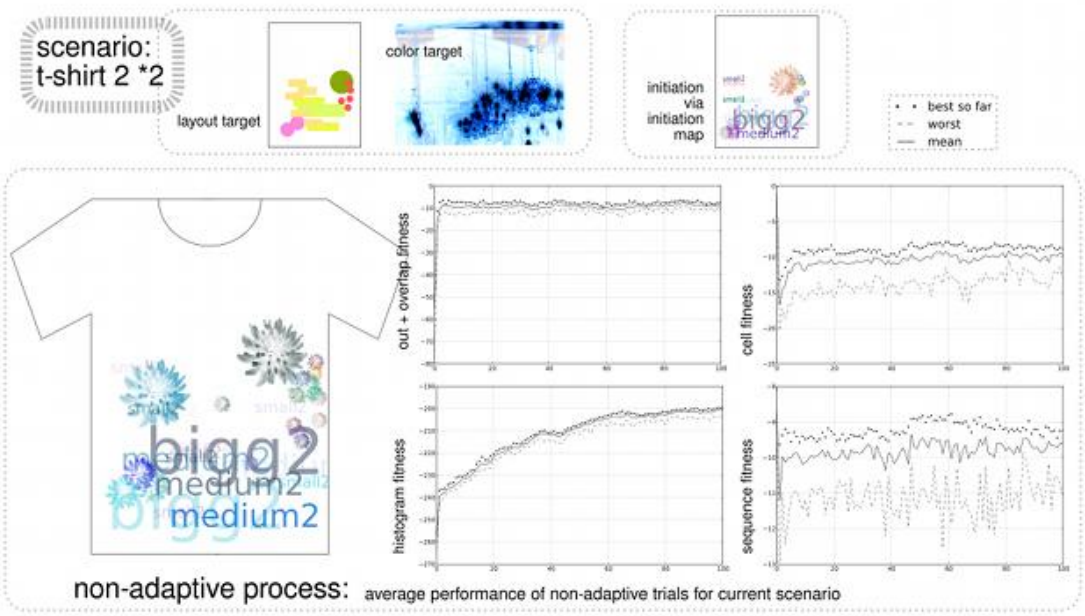


Figure 4.23 : The effect of the initiation maps.

In addition to its previously stated drawbacks, there are also advantages of the priority list approach. It provides the user with a control mechanism over the evolutionary process. As mentioned above, in the example scenarios, a fair degree of overlap was desired, hence a fitness level between -10 to -5 was desirable for the Out + Overlap objective, which can be controlled with these thresholds. These values also provide a control mechanism for the allotment of the computation time to different objectives. In Figure 4.22, it can be seen that the Pareto-based version distributes its resources evenly throughout the process and tries to improve all fitnesses as much as possible. On the other hand, in the example in Figure 4.23, the naive Interleaved EA first gives priority to the layout parameters, and then switches to the Color objective and spends most of its time for color adjustment. This was a deliberate setting, because the color component was assumed to be the most influential feature for visual evaluation.

4.1.9 Conclusions and an evolutionary collaboration model

In the preceding sections, an evolutionary approach for generating graphical arrangements was described, an image-based preference setting interface and a similarity based evaluation approach were exemplified, and a kind of multi-objective EA, the naive Interleaved EA was demonstrated. Coupled with the design_proxy, the interleaved EA is a decomposition and collaboration approach. In its self-adaptive variant, it becomes a dynamic, multi-level, parallel EA.

The experimentations in the previous chapters were mainly aimed at illustrating the technical aspects of the approach. Through a series of design experiments, several aspects of the approach have been examined. Overall, it can be claimed that the experimentations demonstrated the applicability of the approaches. The specific questions that were asked to evaluate the approaches concerned the success of the objectives in single and multi-objective settings, the operation of the naive Interleaved EA for the scenarios, the viability of the adaptivity schemes for parameter control, and the versatility of the adopted approaches.

Through the trials, the objectives were shown to work reliably and to carry out assigned tasks as expected, in both single-objective and multi-objective cases, which also demonstrates the applicability of the naive Interleaved EA. This last issue was further demonstrated by a comparison with a Pareto-based approach. For the adjustment of other process parameters, the self-adaptive variant of the naive Interleaved EA was shown to be a viable option for poor initial parameter settings, and at least as well as the non-adaptive version for the best parameters. Finally, the versatility of the approaches was demonstrated by their application for twelve variants of three different scenarios.

It should be noted that, within the implemented system, the mechanisms for evolving the layouts are rather primitive, and they are proposed only for illustrating and exemplifying the proposed approaches. On the other hand, although simple, the color evolution mechanism is able to attain subtle results. Considering the fine-grained distribution of color gradients within an image and the corresponding difficulty in manually creating such a distribution, it can be claimed that even human designers could benefit from the implemented color evolution mechanism.

One essential objective of the presented approaches was to carry out the visual evaluation operations automatically. In most evolutionary design applications, either the aesthetic dimension is ignored or it is relegated to the human designers within interactive processes. The aim of the approaches presented in this chapter was to exemplify an alternative method, where the human user is freed from the tiresome interactive process. The image-based interface has been proposed for easily capturing the preferences of the human designers before the evolutionary process starts. Through the trials, the image-based interface was shown to assist in easing the burden of visual preference definition.

An important aspect of the image-based interface is the separation of the objectives. Because they concern different objective types, the target images for layout features and color distributions can be different. In other words, as illustrated through the applications of the approach, it is possible to generate new images using different targets for different aspects of an image. The resulting images are considerably different from the targets, yet, their specific aspects resemble these targets. With this approach, it is possible to generate a variety of new images using existing images as targets. In the next chapter, this separation of the definition procedures for different types of objectives will be utilized for separately defining functional and formal aspects of plan layouts.

Furthermore, target images can be collected within image pools, to be used later for evolutionary design processes. In this way, the target definition phase can be separated from the evolutionary process. Indeed, the target definition process may become a continuous occupation for a collection of agents. For each image feature, there may be a pool of target images, which may further be grouped into “style-sets”. Following this line of thinking, a collaboration model is presented in Figure 4.24, which is compatible with the Interleaved EA.

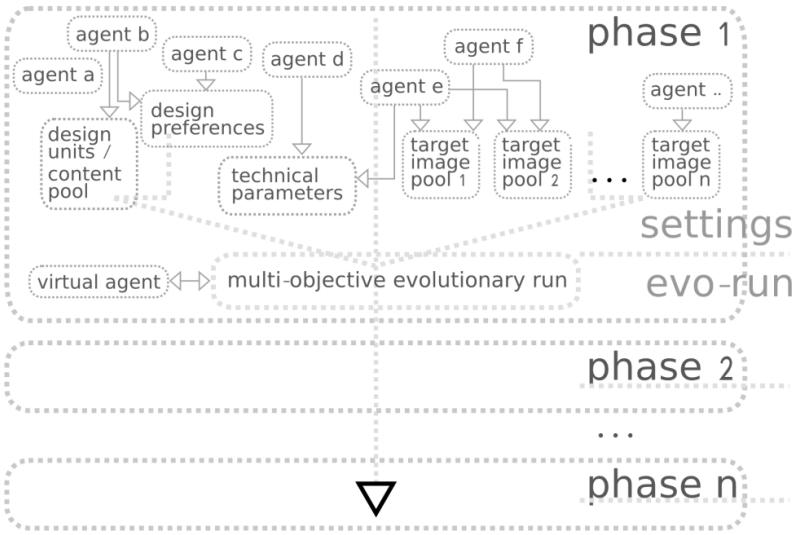


Figure 4.24 : Collaboration model for Interleaved EA.

This model can be seen as a kind of integral decomposition approach. As can be seen in Figure 4.24, there may be distinct tasks within an evolutionary design process. These tasks may concern, for example, production of a background image, production of the schematic layout, insertion of the texts, etc. If we descend one level

down, within each task layer, problem-setting procedures and evolutionary process may be separated. At this level, the evolutionary procedure may be a complex EA, such as the Interleaved EA. When we descend one more level, each of the procedures for setting the parameters of the process may be separated from the others. At this level, a multitude of agents may collaborate with each other for the definition of separate aspects of the process. Furthermore, as mentioned above, this phase can be considered as an independent and continuous process. A user team may define a specific problem, and at the mean time may benefit from ready-made specifications of an EA expert. They can also use existing style-sets for their aims.

It should also be noted that, within this collaboration scheme, it is straightforward for humans to collaborate with artificial agents. Indeed such a system itself may become a provider of target definitions.

Finally, an important point is the usage of domain-specific knowledge within EC. There are several ways how such knowledge could be helpful in evolutionary design. The objective functions are the regular method for knowledge insertion within EC. In this sense, the image-based interface provides a fast method for designers to apply their knowledge. As discussed through the above experimentations, for each of the objectives, there is a different appropriate combination of mutation procedures, selection operators, and parameters. Domain-specific knowledge could be used for the determination of these.

4.2 Situating Evolutionary Computation within Architectural Design: Architectural Stem Cells Framework and design_proxy.layout

The previous chapters and sections concerned the development of an evolutionary design approach (design_proxy), an evolutionary design platform based on this approach (Interleaved EA), and an application through this platform (design_proxy.graphics). This section will continue on this path to further develop design_proxy for a real world problem, i.e., architectural layout design, through the introduction and testing of design_proxy.layout (d_p.layout). This task will be carried out by starting from the most general level. In this case, the specific task of an evolutionary system has to be defined and situated within the general context of architectural design, starting from an overall level, gradually moving downwards from scale to scale while proposing structuring mechanisms for each level. These

proposals for structuring the vaguely delimited field of architectural design will come out as an overall research agenda, which can set forth new paths and potentials for future studies.

In the beginning of the section, it will be examined how architectural design in its totality could be subjected to a treatment by automated agents and evolutionary approaches. This consideration will allow us to situate the d_p.layout system within the general context of architectural design through two complementary mechanisms, layer-based decomposition approaches and the Architectural Stem Cells (ASC) Framework. The structuring attempted with these proposals is not based on temporal progression or phases, but rather on differing tasks. Because these tasks are dynamically intermingled in real design processes, they can only be revealed in posterior analysis. Therefore, decomposition into layers, which is carried out in isolation, is purely fictional and static. It only becomes useful when it starts to include a dynamic integration scheme for the same layers, hence the ASC proposal. Thus, after the introduction of a decomposition scheme for architectural design, the ASC approach will be described and illustrated. The ASC Framework's short and long-term goals and potentials will be discussed through a fictional ASC-based artificial design agent.

The layout task will be located at this point and an introductory review of EC studies on architectural layout generation will be given. After this introduction, a revised three-layered definition for the layout problem will be introduced, which allows a relaxation of the layout problem, so that a versatile layout assistant becomes possible with its flexible representation approach and operators, i.e., the d_p.layout.

In the following sub-sections, task definition, problem representation, and the procedures for initiation, selection, variation, and evaluation will be described and illustrated. Additionally, revisions and additions over the graphics-based interface and the evolutionary system (Interleaved EA), which have been mostly inherited from the previous d_p.graphics application, will be described, and the similarity-based evaluation approach will be discussed more thoroughly.

Detailed experimentations have been carried out for verifying the correct operation of the system and its constituents. These will be described in sub-sections from 4.2.6 to 4.2.8, which will concern the operation of the objective functions, the effect of

adaptive parameter control, and the comparison of the Interleaved EA with a regular rank-based approach. As part of the multi-objective test series, an example multi-floor trial will also be illustrated. In the remaining sub-sections, considerations on the utility of the system's overall and specific aspects will be assessed within the context of two workshop applications.

4.2.1 Problem Structuring and Architectural Stem Cells Framework

Architecture is an extremely complicated field and the question regarding where and how to apply EC techniques within architecture is an undefined one. On an overall level, previous discussions within this thesis concerning the application of EC within design fields apply to architecture as well. However, on a more specific level, i.e., regarding the application of these approaches in architecture, potential directions are infinite. The first task is, then, to give a practical structure to the above-mentioned question. The exploration in this chapter will focus on devising a single viable reusable strategy to bridge between the overall conceptions and specific applications.

With regard to the task structure given in Chapter 3 (Figure 3.2) specific EC applications require a problem-setting phase. Consequently, exploration for architectural design will start with the problem of problem-setting. Dynamic development of a problem structure involves dynamic decomposition and re-integration strategies. Figure 4.25 gives an overall hierarchical decomposition scheme for architecture (with reference to Figure 2.7, levels of design activities, Lawson and Dorst, 2009). Further decomposition can take place on each level given in Figure 4.25.

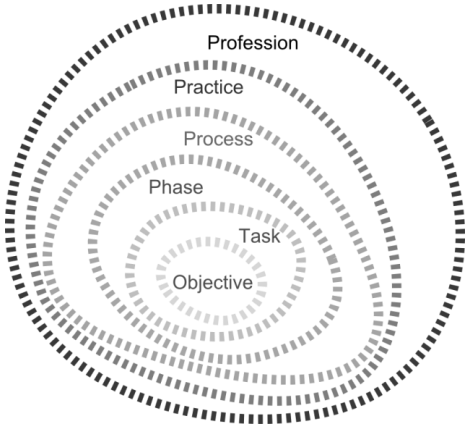


Figure 4.25 : Levels for a hierarchical decomposition of architecture.

On the profession level, architecture concerns all kinds of financial, political, cultural, and inter-personal issues that are related to architecture in general. A sub-area of the profession is the specific actions related to the design of a building. The practice level concerns the customs of a specific architectural firm or team. The process level denotes the level of specific design processes. One step below, there are the phases of a design process, which may not always be discernible. In each design process, there is a variety of types of tasks, and each task includes several objectives. Within this overall scheme, evolutionary design processes will most likely be located on the task level, and each evolutionary process can be further decomposed into a series of objectives.

In the following pages, a conceptual framework will be proposed for structuring an architectural design process. In its totality, this framework will correspond to a phase in Figure 4.25, and it will be further decomposed into tasks with a layer-based decomposition approach (Figure 4.26). It should be noted that the layers of the framework do not necessarily correspond to design stages. In other words, they do not have to succeed each other on a temporal basis. Each layer may comprise a multitude of sub-tasks. These sub-tasks may exhibit dynamic interrelationships with the tasks that are located on other layers as illustrated in Figure 4.26.

Figure 4.26 : A layer-based task decomposition approach.

The layer based decomposition approach has been loosely based on the horizontal decomposition scheme of Rodney Brooks (Figure 4.27). In a seminal study, Brooks (1999, originally published in 1986) proposed to convert from the temporal hierarchical decomposition of the then dominant AI approaches to a parallel yet still hierarchical architecture. Instead of breaking the process into a chain of information processing modules as in Figure 4.27a, Brooks proposed a decomposition in terms of

behavior generating modules, each of which connects sensing to action (Figure 4.27b). The scheme is still hierarchical in the sense that layers are added incrementally, and newer layers depend on earlier layers (Brooks, 1999). While brooks' decomposition model is rather concrete with an aimed robotic structure, ours is deliberately kept abstract. Although our model will also be elaborated into a decomposition and integration scheme for architecture, it will not be carried over to an architectural application with multiple layers. The application will integrate several layers into a single problem definition, which will not be claimed to test for the real significance of the approach. This necessitated the open-ended description of the idea on a rather overall level, with an attempt to set forth a new research framework while situating our applications over this framework.

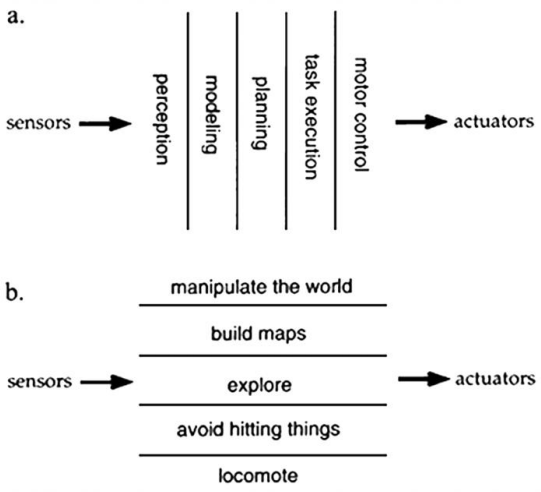


Figure 4.27 : (a) A traditional decomposition of a mobile robot control system into functional modules, (b) A decomposition of a mobile robot control system based on task achieving behaviors (Brooks, 1999).

To move one step further, the abstract scheme in Figure 4.26 has to be replaced by a more concrete one that concerns architecture. The key to this conversion is to find lasting lines of differentiation within architectural processes. There are already well-established separations within architectural design, although these should be considered with caution. It is possible to discern these separations only through posterior examinations of progressions, which are chaotic in actuality (compare with the idea behind the analysis – synthesis – evaluation model of Lawson in Figure 2.4). Consequently, each decomposition scheme will offer a post-facto interpretation, an imperfect generalization.

In a particular building design process, several task layers could be discerned as site organization, basic building massing, relationship with the urban context, functional distribution of the plan units, schematic and detailed drawings for this functional distribution, details for building systems, structural design, and so on. Layers might have sub-divisions in the form of different scales and different productions. Sub-layers of each layer could be initiated at specific moments during the overall process, and the final moments where they would yield their outputs could be determined. At any time during the process, several layers could continue onwards together. The procedures that would be related to a specific layer could undergo modifications whenever they are necessitated due to a change in another layer, or because of its own progression.

Before attempting at a more concrete proposal, several questions have to be brought forward. The first question is, "would it be a viable approach to give a premade structure to specific temporal interactions between layers and operations?" In one respect, this can be seen as the structuring of the simultaneity of layers. However, on a lower level, this structure would result in a partial temporal rigidity. We will return to the first question after detailing the second one: "can we understand this structure in terms of style?"

Style involves a kind of process structuring. It contains roughly fixed patterns for temporal integration of problems and solutions. As a well known example, consider the style of the "Guggenheim Bilbao Museum", by the world renowned architect Frank Gehry, which started to dominate not only the visual appearance of his later buildings, but also the design process of his office, as documented in the 2006 film, "Sketches of Frank Gehry" (Pollack, 2006). In this sense, style involves a reusable virtuality. It contains potential lines of development for new situations, for new scenarios. A style structures the temporal development of the extremely complicated process of architectural design and application. With its virtual structures, style makes designer teams productive. Without style, each move would be primordial; each move would have to be created anew. In addition, style serves as a plane of communication. With its prefixed structures, it enables more effective coordination. Yet, these do not remove the need to deploy an immense amount of intellectual effort for each new architectural situation. There is repetition in style, in both process and product dimension, yet this repetition is by no means simple. Relationships between

the packaged solutions and problems of a style have to be revised for each new situation. Thus, style has to be flexible and is essentially in need of interpretation; though it is not necessarily abstract. Some aspects of style, such as the appearance and texture of the folded metal cladding of Gehry buildings are quite tangible. Yet the production schema of folded cladding is abstract enough to be adapted to new problems, new materials, and new technologies.

In brief, it is clear that there are complex prefabricated problem and solution definitions that are in use in architectural practice, which partially answers our first question above. To some extent, these definitions determine the way to move forward, the way to produce and solve problems, and the resulting product. They involve complicated integration schemas for various layers and tasks and are reusable.

With regard to layer based task decomposition, as an example that strongly interests this thesis, the development of plan layouts can be regarded as a separate task level. It can be assumed that layout development strategies would be integrated into a production process that is roughly defined by a specific style. Within the layout generation layer, a common layout development technique involves the iterative retracing of plan outlines over tracing paper layers, which is a clear example of an abstract reusable strategy that mostly concerns procedure. However, generation of specific plans would be strongly dependent on specific situations and solution styles. The guidance of style with its premade strategies and techniques would dominate both the procedure and the product. This is to say, there would be a strong mutual correspondence between techniques and results and each new technique, be it human or artificial, would influence a stylistic effect in the product. With a broader view, each new method will impose (or will be part of) a specific style and this should be carefully considered when developing new methods. Just as human designers utilize style, artificial design agents, if ever developed, will have to put their own collection of styles in use.

It is clear that while developing plan layouts, other task layers of the overall problem are also being traversed, such as spatial perception, massing, air conditioning, etc. Indeed the layout problem is related in some way to almost all task layers of architectural design. Therefore, establishing static lines of separation is not sufficient to handle a design process; dynamic decomposition and re-integration is the primary

problem here: integration of the layers and operations within temporal progression. This is an essentially open problem as we have discussed throughout the second chapter.

Following these considerations, as a first step forward, a decomposition scheme is proposed for conceptual architectural design (Figure 4.28). When presented with such a structure, architects tend to understand it as if it was the general scheme for all architectural design. Therefore, it appears necessary to clearly state that this structure is not proposed as the general scheme of architectural design. It is only one particular structuring act that is needed for locating and defining a potential evolutionary design assistant, which targets a specific task within architectural design. In other words, it will be used in deciding where an evolutionary approach could be utilized during conceptual architectural design. As stated above, the basic requirement for such an approach is flexibility, which would enable the scheme to be used as a ready-made decomposition scheme for different design scenarios. Therefore, the scheme is kept concise and open to interpretation.

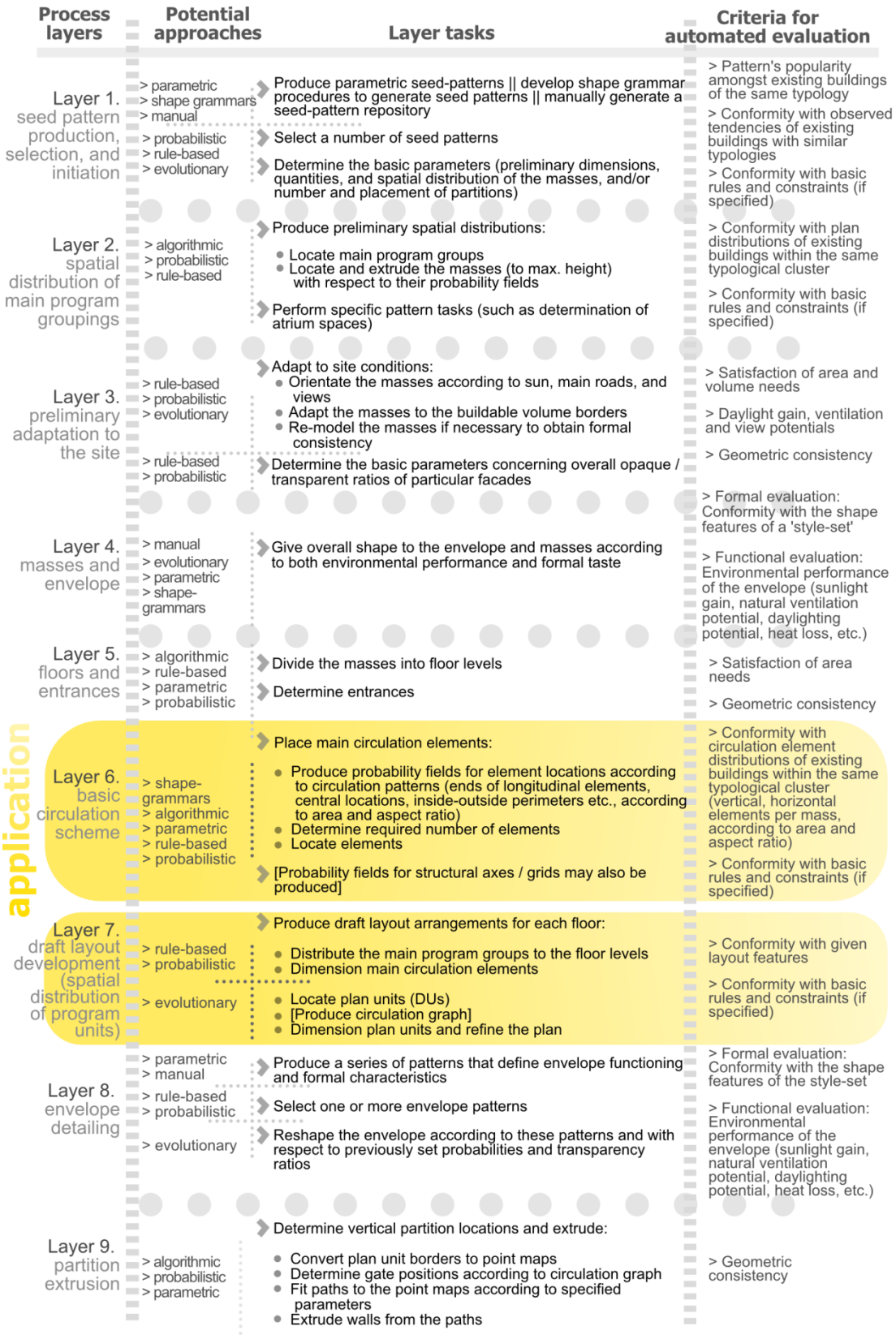


Figure 4.28 : A layer based task decomposition scheme for conceptual architectural design.

In Figure 4.28, each layer concerns a design task, which is composed by a series of sub-tasks. Each of these tasks requires a specific set of methods and capabilities on the side of the designer. Each layer is defined by several of these tasks. A series of potential automated approaches that may be used for handling these tasks are attached to each layer (within "potential approaches" column). Some of these approaches have already been tested for such tasks, though these studies are usually experimental. Additionally, potential evaluation criteria are also indicated.

The architectural problem is structured as the development of schematic layout, massing, and envelope specifications for a specific building program, with specific area requirements, and in a specific urban and legal context. The requirements for these tasks comprise the quality of the solutions with regard to a set of potential evaluation paths, as well as the minimization of the time spent on the task. Yet, the objectives do not have to be exactly specified before the process begins.

The quality of the solutions comprises formal qualities such as the mass formation and the facades, which are linked with both functional and aesthetic goals. Functional goals include spatial distribution and dimensioning of the plan units (such as rooms, entrances, and circulation elements) with regard to orientation, access, and other contextual conditions. There may be other performance requirements concerning circulation cost, natural illumination, sunlight gain, insulation levels, natural ventilation etc. However, it should be noted that, at this early stage of architectural design, a strict application of performance requirements could be detrimental to the design process. Therefore, dynamic, flexible, and tolerant evaluation approaches should be employed for these stages.

A few of the proposed task layers concern rather straightforward tasks; however, the majority of them concern complicated processes, which have to be further decomposed. For further decomposition, a further structuring of the situation is necessary. However, this structuring cannot be achieved at a single task layer in isolation. Rather it should be worked out together with the integration scheme of all layers. Therefore, our second step towards a structuring proposal will be an example strategy for integrating separate task layers of the framework through a solution-based approach.

For this aim, the architectural design process will be envisioned as the development of an Architectural Stem Cell (ASC). Most probably, Manuel de Landa's "abstract building" notion is the closest predecessor of the ASC idea. Referring to Deleuze and Guattari's "abstract machine" concept, de Landa (2004) recommends the employment of topological invariants just as in the evolution of vertebrates in order to prevent the evolutionary process to become a random search. These are the "body plans" that would give structure to the evolution of architectural artifacts. ASC does not propose to keep a constant topology throughout evolution, yet the notion of abstract building conforms to what is envisaged with the ASC proposal.

The ASC idea has several other precursors, including the notion of "concept-seeding" proposed by John Frazer (1995), and the related "schema-coding" approach proposed by Patrick Janssen (2006). The approach of Patrick Janssen has been an important overall inspiration for this thesis; in particular, the schema-coding approach, which divides the design process into two phases, where a problem-setting phase would be followed by the evolutionary solution process. Another precursor is the idea of "patterns", which is introduced by Christopher Alexander (1977).

Within the context of this thesis, an ASC is envisioned as a holistic solution proposal, which would be defined by a series of tendencies and affordances. It should be developed gradually, yet not only within the limits of its affordances, but also according to the needs of a specific context. When it is first initiated, its formation would be undetailed, yet it would exhibit a totality. Through the demands and problems that are manifested within the layers of a task, each layer of a stem cell will start to gain shape and detail. However, even in its most abstract status, it would remain an architectural proposal, which could be interpreted by a human agent.

With an alternative viewpoint, an ASC is a prefabricated architectural design strategy. The tendencies and affordances of a stem cell would be an embodiment of the knowledge of the field of architecture. In this regard, the idea can be compared with typologies. A well-known building typology could be used as a template in the development of a stem cell.

An ASC can also be envisioned as a parametric definition. Consider the abstract three-dimensional formation in Figure 4.29. The divisions within the prismatic mass may correspond to a variety of spatial definitions. Each division may be developed as

a kind of functional unit within a building. Alternatively, a division may be chosen as a gallery or as a circulation tower. The tendencies and affordances are necessary to limit the number of divisions, or to define the potential roles of each division with respect to a specific context. This can be likened to a biological stem cell, which develops differently according to its location within a body, hence the phrase Architectural Stem Cell.

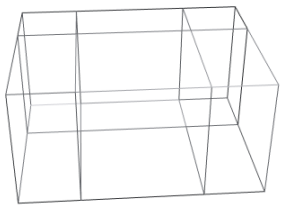


Figure 4.29 : An abstract 3D formation as an architectural stem cell.

The ASC approach is developed in combination with the task decomposition scheme in Figure 4.28. An ASC's development process may be defined and controlled by the layers approach. Three such ASCs are proposed and illustrated in Figure 4.30. The development process is carried out manually, with the purpose of illustrating the approach. In Figure 4.30, the unfolding of the ASCs pursues a regular progression, starting with the first layer, continuing downwards, although this is not a prerequisite of the approach. It should be noted that the progression of the tasks could be organized differently as illustrated in Figure 4.26, i.e., by using feedback between interdependent layers.

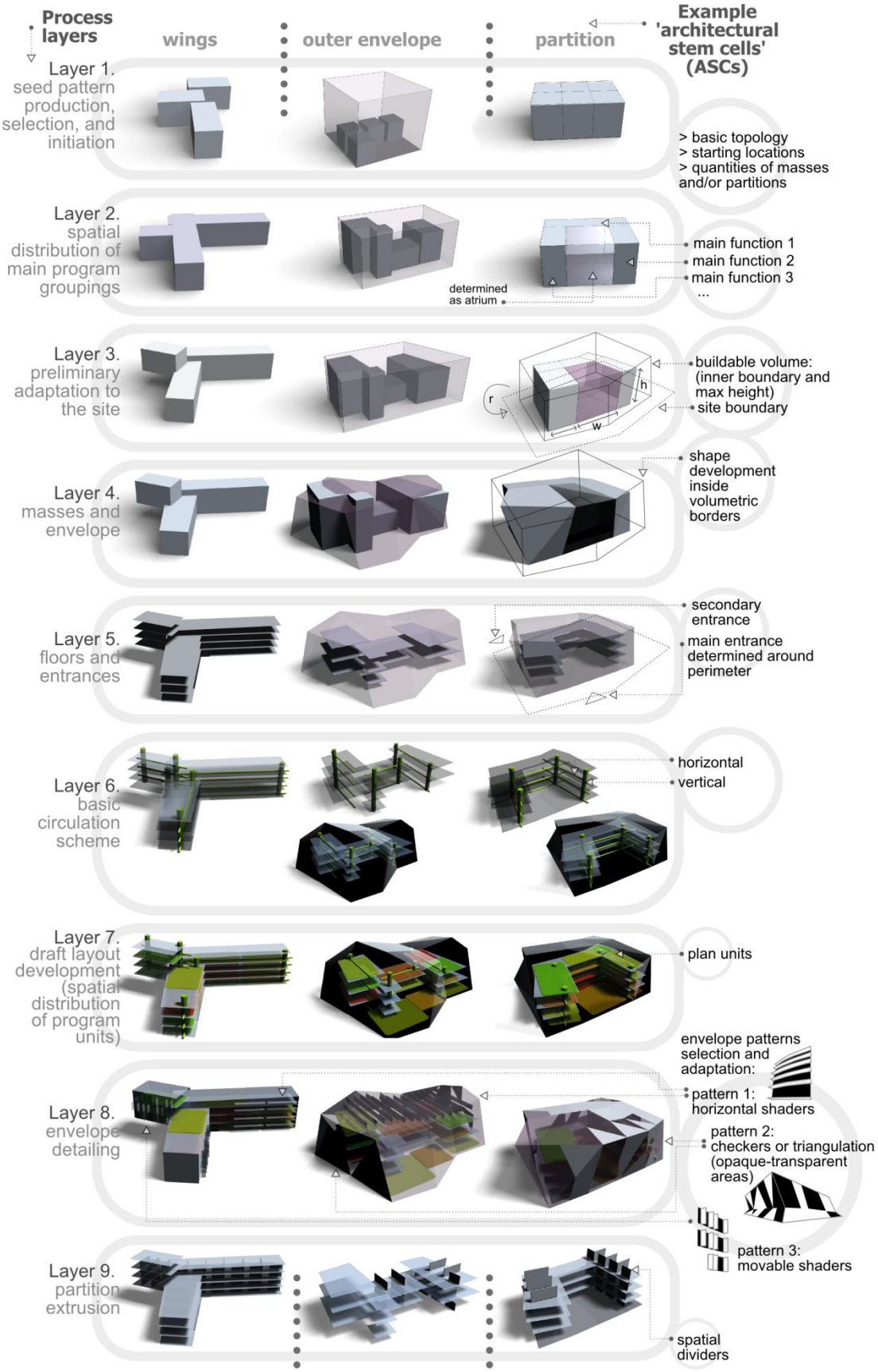


Figure 4.30 : Three example ASCs, manually developed within the task-based decomposition framework.

Together, the layer based decomposition approach and the specific ASCs constitute, first, a structuring framework for architectural design, and secondly, a research framework. The task decomposition scheme has been put to use as a working template to define and locate an example EC system, i.e., design_proxy.layout (d_p.layout). The framework readily defines several other tasks that could be entrusted to EC systems within a design process, like building massing, basic circulation, and facade studies. These layers could be combined for multi-layer systems. Thus, in the short term, such a framework can help in coordinating a research agenda.

Another important question concerns long-term studies: "what contribution can be expected in the long run from such structuring frameworks and ASCs, both in practice and research?" An initial answer can be that, specific ASCs may be understood as preliminary automation frameworks. In the long term, such frameworks could be used in the development of design-specific artificial agents. We will discuss some such potentialities through a fictional ASC-based artificial design agent.

An ASC-based design agent would function through prefabricated styles of its own. This requires contextual interpretation in addition to design capabilities. Although artificial general intelligence is not available, this should not prevent us in our quest for design-specific production methods. This line of research can be pursued with the hope to bring all separate pieces together around a prospective centerpiece, i.e., general daily intelligence. If ever such intelligence becomes available, all the design specific aspects will gain mutual meaning. Because, just as design-specific techniques do not work competently without daily intelligence, a general daily intelligence on its own would not be sufficient to design. Design is an expertise, which exploits daily intelligence. This expertise appears to be embodied within collected and interpreted examples as well as production techniques. This is the reason why the ASC idea has been brought forward to continue developing, collecting, and interlinking such design specific techniques.

As has been discussed in the preceding chapters with respect to Negroponte's research program, artificial design intelligence can be developed only within processes where humans and machines cooperate, which would also be valid for an ASC-based agent. It appears clear that, in the short term, an ASC-based agent will

not be as successful as human designers. Then the question becomes, while humans are capable of designing much better than the artificial systems, how will human designers accept such systems in their design processes? Could an ASC-based agent, even in its primitive stages, help its human counterparts to design better, easier, more pleasurably, more productively, more rigorously, or with a broader perspective? Or, would it simply demand the human to limit her sensitivity artificially? These questions reveal the roadmap to move forward.

An ASC-based agent needs to be a pleasurable and productive companion to its human user. When it comes to productivity, within the state of artificial design intelligence, an ASC-based agent's products would not be competent, yet could be fairly detailed. Therefore, such a system can be practically useful by providing a range of tentative alternative solutions to be worked out more sensitively by the human user. These tentative solutions would function as drafts. The benefit would be an ability to pursue more lines of alternative paths and much faster, although with less intelligence and sensitivity. It could be likened to working with less resolution or with thumbnail pictures.

Consider an ASC-based agent producing a building with its massing, facades, preliminary layouts, basic circulation, etc., from a situation where basic problem (typology, area, cost, etc.) and contextual information (site, neighboring buildings, orientations, access roads, etc.) are given. This would mean producing a series of different tentative proposal buildings within different stylistic characteristics. It would also enable a designer to delve into paths that she would not normally explore just out of a lack of time. An office could carry out its own in-office competitions with such a system at hand where the results could at the least function as the products of a very detailed brainstorming session. Therefore, even with a set of incompetent artificial design techniques, it seems viable that an ASC-based agent could be accepted in an office's workflow. However, given its technical limitations, issues like ease of use and pleasure would still be decisive in an ASC-based agent's practical success. Variety and adaptability of specific ASCs, level of development of techniques, and effectivity of interfaces in communicating information and preferences would also be important. An ASC-based agent should be able to work through several layers within a design process on its own, even if it achieves limited success. At the same time, the human user should be able to intervene whenever she

pleases. By enabling selective use at different stages of a design process, flexible usage should be made possible. Then, by bridging between the singular uses of an ASC-based agent, human users could weave a whole design process. Some of these issues will be exemplified and discussed through the d_p.layout system.

As a last remark, the potential role of Evolutionary Computation (EC) in this research framework will be briefly discussed. Which of the tasks and sub-tasks defined within current ASC schemes could be handled with EC approaches? Is EC suitable for all tasks within these ASC schemes? These are open questions, which are difficult to answer without specific applications. Nevertheless, as will be discussed with regard to the d_p.layout application, it can be claimed that, whenever the complexity of a task increases, the final success of an EC application tends to diminish, as merely simplifying problem definitions does not appear as a viable option for design tasks. Nevertheless, EC might serve for bridging separate layers by using a series of populations in hierarchical EC schemes. Therefore, producing and adapting hierarchical ECs on the run could be an interesting approach to follow. Our Interleaved EA has such an adaptive scheme, yet a primitive one.

As a result, the ASC idea sets us onto such a long term research perspective, while practically enabling us to move forward by bridging between overall considerations and practical applications. The long term perspective will be left for future research. In the following sections, we will move down to practical applications. As an example application of EC within the above presented architectural problem structure, and in congruity with the ASC approach, a layout generation system has been developed, which is named "d_p.layout". The system undertakes the development of an architectural layout, together with the placement of circulatory elements. As such, it combines layers 6 and 7 in the decomposition scheme (Figure 4.28) and it can be used for all example ASCs (Figure 4.30).

4.2.2 Layout problem and a review of related studies

Plan layout design (PLD) is one of the primary tasks in architectural design processes. It is concerned with the topological and geometrical assignments of activities to space such that, a set of criteria are met and/or some objectives are optimized. Topological space planning may result in a relationship graph, while

dimensioning of the space elements involves producing the geometric properties of the plan (Ligget, 2000; Jo and Gero, 1998).

A typical architectural layout problem contains design units (DUs), relationships between these units, and a context. DUs correspond to spatial wholes that represent an architectural use of function, while relationships can be any attribute that specifies how units relate to each other. Unary properties may also be defined. The context may involve a site, boundaries, terrain, orientation, cardinal directions, climatic data, views, occlusions, and other important features around the site.

A series of design systems directed towards architectural layout problems have been developed, including HeGeL (Akin, Dave, and Pithavadian, 1992), WRIGHT (Baykan and Fox, 1997), and SEED-Layout (Flemming and Woodbury, 1995). These systems operate under various constraints that comprise access, natural light, and privacy as well as spatial, topological, functional, and geometric constraints.

Architectural layout problems inherently comprise multiple and sometimes conflicting criteria. As the number of DUs increase, the search space for solutions enlarges exponentially. The high number of constraints, functional requirements and performance criteria to be formulated and computed, together with the high number of DUs in real life problems render the search and evaluation procedures expensive in terms of computational resources (Jo and Gero, 1998). Therefore, systematic search approaches are not practical for complex real life problems (Ligget, 2000). Against the problem of combinatorial explosion, another line of research formulated the layout problem as an evolutionary search, instead of an exhaustive one. In these cases, EC is basically taken as a generate and test procedure, where a relatively small section of all possible solutions are traversed.

Damski and Gero (1997) presented a system to produce space layout topologies for architectural plans, where layout specification is defined as a set of topological and directional constraints. Fitness functions were based on three topological criteria, i.e., connection, overlapping, and adjacency. Jo and Gero (1998) re-formulated an office layout planning problem. An overall cost measure was used as the fitness function, which considers interactions between pairs of activities and the distance between their locations. The interaction matrix was hand-coded, based on subjective judgments of the client. Gero and Kazakov (1998) updated this study. The cost of

assigning an element, a measure of interaction between elements, and a measure of distance between elements were used as fitness criteria.

Rosenman (1996) explored plan design for houses on several levels. He experimented with several fitness criteria in hierarchical evolution processes. These criteria include perimeter-to-area ratio and the number of angles at the room level, and adjacency requirements between rooms at zone and house levels. In addition to quantitative assessments, qualitative assessments were made interactively by the user. Rosenman and Saunders (2003) explored fitness functions that compare area, length-to-width ratio, connectivity of rooms, and perimeter-to-area ratios of houses. Due to practical problems, most studies are constrained to a small number of fitness criteria. This means, many important characteristics of an architectural layout are usually omitted (e.g., views, solar orientation, etc.). Another shortcoming is that, while visual and spatial perception is an important component of architectural design, in these studies it is mostly either omitted or relegated to interactive evaluation, which puts the burden back on the human designers.

Michalek, Choudhary, and Papalambros (2002) used a relatively higher number of criteria while separating the layout task into two phases; the first phase aimed at the generation of architectural layout topologies through EC, the second phase concerned the development of geometrical arrangements for given topologies through gradient-based algorithms. Topology phase objectives comprised connectivity, DU overlap, direct circulation path between DUs, and planarity. In the geometry generation phase, constraints functioned for prohibiting overlaps, providing access ways between spaces, forcing units to plan boundaries, and providing minimum space ratios, minimum construction material costs, feasible window sizes and minimum natural lighting. Additional objective functions measured heating cost, cooling cost, lighting cost, wasted space, and access ways, which were unified through a weighted sum approach.

In more recent studies, Wong and Chang (2009) represented adjacencies of DUs in graphs and evaluated the fitness of these graphs in terms of adjacency preferences, budget, and design constraints. Inoue and Takagi (2009) optimized given initial layouts using a pareto-based EA. Evaluations were based on given room dimensions and regularity of room shapes, circulation relations, legal window sizes, wall lengths (as a cost estimate) and plumbing lengths. Flack and Ross (2013) compared

weighted, pareto-based, and rank-based approaches in multi-objective EC. Evaluation constraints included max.-min. numbers, sizes, areas, ratios, and types of rooms, having windows, proximity (connectivity) between rooms, and area comparisons between types of rooms. A criterion measured the graph-depth average of the distance between the main entrance and each room. Additionally, a measure for closeness to "ideal geometric ratios" was included. Finally, Rodrigues, Gaspar, and Gomes (2013a, 2013b, 2013c) developed a very detailed hybrid evolutionary technique, which couples Evolutionary Strategies (ES) with Stochastic Hill Climbing (SHC). Evaluation consisted of a weighted sum of all penalties for not satisfying the user requirements and constraints, which comprise, adjacency, DU overlap, location on floor plan, clear area in front of every opening (for doors and circulation), opening orientation, admissible dimension intervals, building boundary overflow, and compactness.

It should be noted that, most of the above-mentioned evaluation criteria could have been obtained from the formal characteristics of existing buildings. Any existing building exhibits a working collection of embedded choices and holistic solutions. Most of these are reflected through layout representations, which usually contain rich semantic references. The application that will be described in the following sections proposes to derive its criteria from existing buildings. The basic idea is that the layout features of existing buildings can serve as a kind of wholesale evaluation function. In a series of studies, Sean Hanna (2005; 2006; 2007a, 2007b) explored this possibility by using machine-learning techniques to generate implicit objective functions for usage in evolutionary generation of floor layouts.

It can be claimed that, despite more than 40 years of effort, artificial architectural layout generation systems have not reached competence levels that are comparable to human designers' and as a result, architectural layout development is still a human task. One of the reasons for this failure has been a preference for engineering oriented, overly rigorous, hence, overly reductive problem definitions. It appears that before attempting to develop new layout systems, a reconsideration of the layout problem is required; with an orientation specifically towards architectural design. This reconsideration should involve both technical aspects and task definition, in order to reveal the aspects of the problem that could be handled with existing technologies, and also the required innovations.

It should be admitted that, in its architectural variant, the layout problem is an extremely complicated problem, which is in connection with almost all aspects of a building's design. In engineering oriented layout problems, many psychological and spatial aspects of the problem are traditionally omitted, and the problem is handled only in terms of efficiency, effective use of spaces, or cost, which is by no means a sufficient list for architectural layouts. Although the problem has to be converted into manageable constituents due to the complexity of the problem, this should take the form of a structuring instead of a restriction, or overly ambitious simplification.

A superficial examination of human layout design processes would reveal a progression where, first, a rough placement or division of main groupings take place, after which, a gradual effort for partial arrangements of design units would continue as integrated with drawing and dimensioning. During this effort, orientations, contextual and spatial relationships, internal spatial relationships and adjacency, and visual and perceptual issues would be influential as well as the guidance of the overall conceptual, contextual, and stylistic considerations. At some point, refinement, detailing, and optimization would start to become conspicuously dominant over wholesale changes. However, whenever demanded by an aspect or a sub-system of a building, a revision that spans many layers would be required and would be propagated through all layers. This is not to deny that revisions, detailing, and refinement start from the beginning and run along the totality of the process. On the contrary, most of these layers span through most of the process.

In brief, with a post-facto interpretation of human layout generation, a viable separation of the overall layout layer involves three main sub-layers: (1) main circulation and zoning (overall level), (2) patching (arrangement), and (3) refinement (detailing). It should be stressed that these layers do not correspond to temporal distinctions. On the contrary, all three layers exist throughout the process although with differing prominence. These layers can be seen as the three aspects of the same threefold process. Because these sub-layers are dependent and simultaneous, in practice, a parallel solution procedure is required[20].

[20] Rodrigues, Gaspar, and Gomes (2013a, 2013b, 2013c) exemplify such a layering of the layout task by separating the refinement task. The re-integration takes place during the evolutionary process, in which the refinement actions are nested as a secondary search process.

Zoning and main circulation is the overall level, while main operations take place on the patching level according to the zoning and basic circulation decisions. However, patching could also demand a change in the zoning and vice versa. On the other hand, refinement here refers to a more sensitive operation in which expectations of spatial experience and psychological comfort, as well as compositional elegance of the drawn lines would be regarded. This sub-layer mostly concerns formal issues and it could as well need to re-determine the other sub-layers. Therefore, in spite of the apparent hierarchy of the sub-layers, the progression would have to be much more complicated and the interventions would be in both ways, up and down.

According to the above structure, each sub-layer may have its own set of rationales, and within each sub-layer, there may be sub-problems. In the patching layer, orientations, entrances, circulation, and functional relationships would appear as sub-problems, which could be handled through separate objectives, operators, constraints, and rules. In the refinement layer, deciding on the final formation of the walls, doors, and other elements would be amongst the problems and style would be influential as well as rationalization in terms of convenience.

Final drafting of the plans may be excluded from this layering, because once elements are determined with semantic information, drawing can be handled by rule-based and parametric methods. In addition, the question of the temporal progression of the overall layout generation layer amongst the other layers of architectural design should be examined separately. This question would involve the determination the transformations of this threefold layout generation process from conceptual design to production drawing phases.

The threefold problem definition brings forth important advantages. First, as will be seen with the d_p.layout application, it relieves a layout generation system from the requirement of formal rigor and exactitude, which is incongruous with the character of the architectural layout problem especially in the early phases of a design process. Secondly, it removes the need for strict, clearly defined layout proposals, which is far too demanding for available artificial design technologies in the context of architectural evaluation. In short, by enabling us to appreciate the potential place of a more flexible layout representation in the layout generation process, it helps us to reconcile current technology with a practical application. Thus, the first peculiarity of

the d_p.layout is its flexible and permissive layout representation, illustrating the first tenet of the design_proxy approach.

The applications that will be illustrated in this thesis operate within the patching layer. This is the most straightforward implementation and has its obvious drawbacks, i.e., no refinement and no automated resolution of vagueness. However, in principle, it is possible to handle at least two of the sub-layers, i.e., zoning and patching through the d_p.layout, either separately (the latter starts when the former ends), or in parallel (continuous redefinition of the patching problem by zoning and vice versa by progression checks). On the other hand, the refinement layer appears to require other techniques, different representation approaches, and in the worst scenario, a more developed artificial intelligence. Therefore, it cannot be tackled with the current application. However, it will be claimed that the characteristic roughness of the results of d_p.layout renders them more open to interpretation, which becomes an advantage given the low level of intelligence of the system. The d_p.layout system functions only as an assistant and should simply inspire and support interpretation.

The second important issue with d_p.layout is its evaluation approach. As was stated above, d_p.layout uses layout features of existing buildings for the evaluation of new proposals. It achieves this by measuring the similarity of a proposal to a target layout in terms of specific features; hence, it appeals to examples and similarity. At the same time, popular approaches such as adjacency matrices and cost calculations are given up in favor of more tolerant functional evaluations, which may implicitly concern psychological and technical aspects. This approach resulted in, first, a more intuitive visual interface in terms of graphics, and secondly, a more open-ended evaluation approach, which is indefinitely open to development, one that can be continuously enriched with different kinds of evaluations.

4.2.3 design_proxy.layout

design_proxy.layout (d_p.layout) is envisaged as a design assistant that would participate in a regular architectural design process through its ability to generate draft layout arrangements. It supports varying degrees of user control; thus, it can be used as a tool, as an assistant, or as something in between. This flexibility of use brings forward a potential for pleasurable and playful usage.

With a cumbersome user interaction approach, it would be hard for a design tool or assistant to be adopted within a design process. An evolutionary design tool or assistant should interact easily with its user or companion. It should solicit information concerning problem-setting and preferences with a fast and intuitive interface, and should deliver its results in an easily interpretable way. Numerical parameter definition is out of the question for these aims, because designers are not usually able to explicitly state what they know. Moreover, an explosion of parameters arises when complicated problems like architecture are encountered.

As was the case with the d_p.graphics application, the proposed solution for d_p.layout is a graphics based interface, to be used both ways. From the user to the system, user preferences and problem settings are defined through a graphic interface, which registers basic characteristics of candidate layouts. The open source vector graphics application, Inkscape, is used for this graphic interface. Through a layering template, the user is able to define contextual information, floor borders, and optionally a series of fixed and mobile DUs. This constitutes a highly flexible interface, which is easy to use, and which makes it easy to adapt to a wide range of architectural scenarios with different amount and types of DUs, floor numbers, and plan outlines. Furthermore, the approach is indefinitely open to development and revisions.

The same layering template is also used for semantically marking example layouts to feed evaluation information to the system. Through vector painting, parsing, and additional procedures, the example layouts are converted into targets for objective functions. This preparation of the evaluation targets is a separate operation and can be carried out independently.

Although there is a long list of process parameters that operate on the background, the user does not need to bother with these at all. However, it is possible to adjust these parameters as well; d_p.layout gives its user an option to apply differing amounts of control. For example, it is possible to determine a complete list of DUs with their geometries or a limited set of fixed and mobile DUs, or to leave this entirely to the system. It is also possible to manually distribute some or all of these DUs onto different floors or to expect the program to do the distribution.

On the other direction, from the system to the user, the system delivers its draft layouts in the form of either bitmap or vector graphics that require and facilitate interpretation. As such, d_p.layout aims at offering a fast, flexible, and effective interface.

As will be illustrated in the following pages, functional objective procedures of d_p.layout operate by measuring the similarity of a solution candidate to a target in terms of specific features. This evaluation approach utilizes the knowledge embedded within existing building layouts and can be seen as a method for case-based design, where an example layout is used as a precedent or a paradigm case.

Therefore, at least in its functional evaluations, d_p.layout can be claimed to be using the "family resemblance" concept instead of objective functions based on mathematical formulas. Family resemblance and the related "thread" metaphor have been brought forward as an alternative to classes (Wittgenstein, 1999). A class is an all or nothing concept, and is defined by a limited set of traits or definitions. However, many real world classification attempts raise insurmountable difficulties and the class concept is mostly unsatisfactory (Dreyfus, 1972, p. 38). The main problem is the difficulty of manually forming a limited list of required and sufficient traits that would precisely identify and differentiate a class of things. The traits that characterize things are numerous and hard to define. Worse still, exceptions appear everywhere. Compare these difficulties in determining ideal states for buildings, as optimization approaches would demand. Consider also the difficulties that an attempt at a taxonomy of architectural typologies would encounter.

On the other hand, the family resemblance concept works through similarity or proximity to a paradigm case, or a prototype, which does not need to be ideal. From a cognitive science perspective, Lakoff and Johnson (1999) claim that most concepts are not characterized by necessary and sufficient conditions, and prototype-based reasoning constitutes a large portion of actual human reasoning. In general, the structure of concepts includes prototypes of various sorts: typical cases, ideal cases, social stereotypes, salient exemplars, cognitive reference points, end points of graded scales, nightmare cases, and so on (Lakoff and Johnson, 1999, p.77). Each type of prototype uses a distinct form of reasoning.

Here, the thread metaphor helps in explaining how dissimilar items may still be part of the same general group by only partially resembling the ones near them. Like the fibers on a thread none of which runs through the whole length of it, ". . . no two members of a family need have any identical features for them all to share a family resemblance" (Dreyfus, 1972, p. 40).

The family resemblance conception is robust and tolerant, it can work for typical and atypical examples alike, and it simply does not collapse. Just as family resemblance is an alternative to classification through a limited set of traits, similarity-based evaluation can be seen as an alternative to constraint-based evaluation approaches, rule-based state definitions, and approaches that measure a candidate's performance in terms of its relationship to predefined mathematical formulations. Although constraints are useful for practically limiting a search space, it is indeed not possible to positively define a desired state only in terms of constraints. Moreover, it is practically impossible to define or evaluate design states with simple if-then style rules. Such approaches demand the definition of states through limited sets of explicit statements. For this to be possible, the defined states themselves have to exhibit a suitable ontological constitution. This conception corresponds to younger Wittgenstein's rigorous language, which would describe relationships of atomic items (Wittgenstein, 2003). This is exactly what he came to criticize in his later years with his family resemblance conception (Wittgenstein, 1999). This has been a recurring theme in Dreyfus' discussions; he puts forward the contrast between the two paradigms to support his case against AI, which he claims is won by him (Dreyfus, 1972; 1999; Dreyfus, Dreyfus, and Athanasiou, 1986). For practical aims, his conclusions can be summarized as follows: trying to define the world through a limited set of explicit statements is the result of an unsuitable paradigm and it proved practically inconvenient for even simple test domains.

As will be seen, in coding a building's layout arrangements in terms of discrete items, the practical approach that is adopted for the d_p.layout, in effect, follows the above assumption of younger Wittgenstein, concerning the ontological constitution of the constituents of the world as distinct, atomic entities. However, his other assumption, which claims that the factual states can be described and assessed through listing a set of explicit statements that describe the relationships formed between these items,

is found impractical, and the similarity based approach is preferred, as it does not demand explicitation.

As a last remark, the problem with the mathematical objective function approach is that it demands strict definitions. These approaches are suitable for optimization procedures but not for conceptual design, where most of the goals are essentially incongruous with fixed mathematical formulation. It is exactly here that evaluation through similarity should be put to use. With the similarity-based evaluation approach, an objective function would essentially remain incomplete. To tentatively complete its definition, it can use a pool of examples together with statistical techniques. Different objectives can use different example cases at the same evolutionary run. In addition, the evaluation information can be updated simply by adding or modifying example layouts without any updating or recoding on the system, hence information and procedures can be separated. The evaluation approach can also be improved by adding new analysis methods, which can be increased indefinitely. However, it should be noted that there would be a practical limit for the number of simultaneously followed objectives.

As a consequence of the similarity based evaluation approach, it becomes possible to share and reuse evaluative information, in the form of example pools (see Figure 4.24 where a collaboration model is given for this approach). Furthermore, each worked-out example can be tagged in terms of its typology, scale, and contextual properties and shared over an open internet platform[21].

Before passing, a legitimate question can be asked concerning copyright issues with regard to the example buildings. Ultimately, this appears as a political question that can only be decided through a continuous debate between different sides of the problem. However, it should be noted that, neither the architectural solutions, nor parts of the plans are being copied in this approach. Rather, the overall information that is embedded within the plan drawings is interpreted and utilized, as is regular practice within architectural profession. Architects publish their technical drawings and these are examined by other architects. Ideas and information are continuously

[21] Note that with tagging, the problem of classification re-enters the scene on two levels. On the first level, tagging is classifying. And on the second level, using these tags to search for relevant items would require another classifying effort.

being shared this way. Examining and adapting existing examples is the regular practice of architecture.

An example 'painted' target for d_p.layout is given in Figure 4.31. On the right side a collection of vector painting styles are given. Each style corresponds to a separate layer. Using these layers, the system parses the '.svg' document and defines each shape as 'something', such as a zone, a plan boundary, a front, an open facade, a kind of functional DU, a scale line, north arrow, a road, etc. Note that until this point the approach remains within the limits of atomism. The scene comprises discrete items with definite properties. The context of the architectural layout problem allows this kind of discretization, to the extent that it corresponds to the set of items that is being used by human designers.

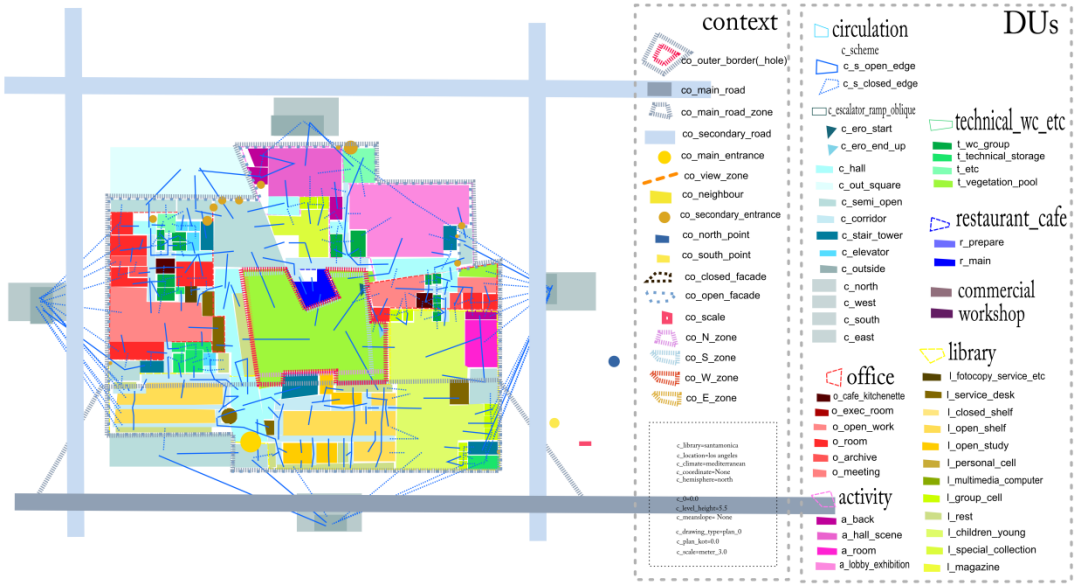


Figure 4.31 : Example target, ground floor of Santa Monica Library by Moore Ruble Yudell Architects. Vector painted on Inkscape according to given layers and swatches (seen on the right).

This vector painting operation is carried out over a given plan drawing. In Figure 4.32 a simpler version can be seen with only the DU areas painted. Although most of the elements are already given as 'something' due to the communicatory character of a technical drawing, the painting operation nevertheless requires interpretation.

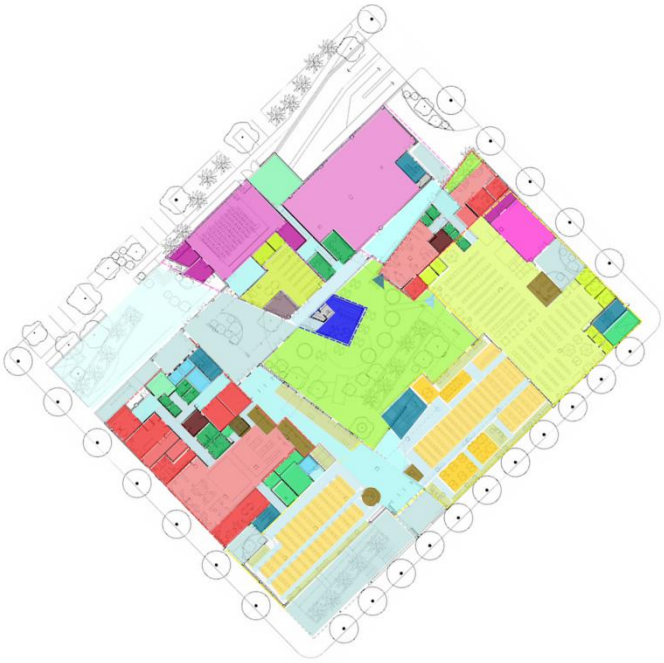


Figure 4.32 : Example target simplified (only DUs), ground floor of Santa Monica Library.

Obviously, there is no perfect discretization and the painting operation involves interpretation with regard to the context. The only viable option here is to move forward and try different options within practice. Consider the reddish areas within the drawing in Figure 4.32. These correspond to different office spaces. The pinkish area in the middle of reds represents the open office area as a single polygon. However, this open office area includes several subareas, like desks, chairs, couches, etc. On the other hand, the orange-yellow areas represent library-related functionalities. Because the test cases were mostly about the library typology, these areas are further separated into desks, rest areas, bookshelves, and circulation areas.

The list of functional DUs is given in Figure 4.33. There are three columns in the figure. In each column, the middle part shows the painting style for visual assessment. On the left side the layering for the painting is given. On the right side the simplified library program that is used for the experimentations is mapped to these primary layers.

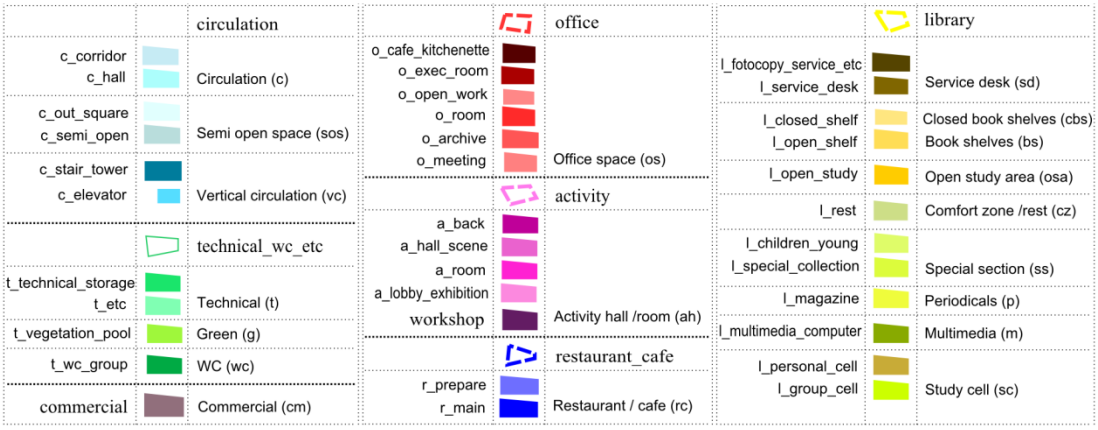


Figure 4.33 : Library functions. In each column, painting layers (left side) is mapped to a simplified library program (right side).

4.2.4 Task definition, representation, initiation, selection, variation, and the evolutionary process

In its most basic expression, the task of the d_p.layout is defined as adequately populating a series of plan borders, i.e., several floors of a building with a fixed set of arbitrary polygonal DUs, whose forms and dimensions are also determined and fixed at the outset (Figure 4.34). Here, the term 'adequately' refers to various considerations in terms of functionalities, performances, formal qualities, and constraints. Therefore, the basic goal is a proper placement and arrangement of DUs with regard to each other and to the given contextual elements and conditions.

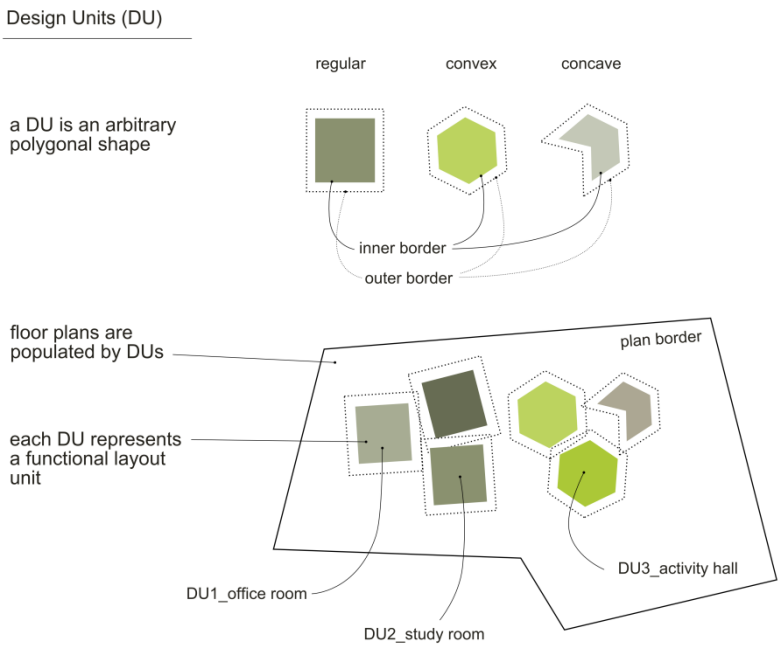


Figure 4.34 : Basics of problem representation.

Although d_p.layout can start its process with a predefined and fixed set of DUs, it can also assume the task to generate this list of DUs by itself, from an analysis of a pool of targets, in which case the task definition becomes, properly determining and arranging a list of DUs over a series of building floors with variable plan boundaries (Figure 4.35). d_p.layout carries out this task with regard to a series of example layouts, and in its current applications it is dependent on well-established typologies like libraries, schools, and museums. If this last point is also included in the task definition, we can state that the task of d_p.layout is properly determining and arranging a list of DUs with regard to a series of example layouts, over specific floors of a building that is within a specific typology.

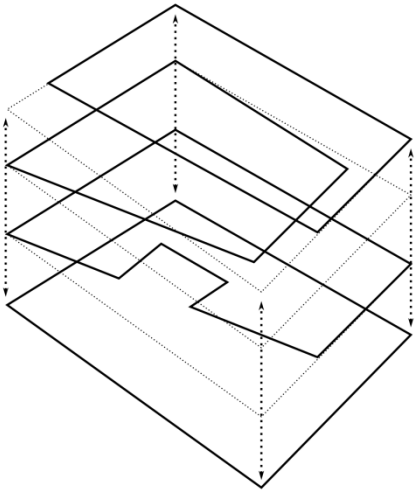


Figure 4.35 : Layout generation for multiple floors with arbitrary outlines.

In principle, owing to the flexibility of this representation, the list of DUs does not have to be fixed and DU dimensions and forms could have been dynamic as well. Such dynamism could be used for implementing a refinement functionality to the d_p.layout system, but this possibility has not been pursued in the current application.

The initiation and representation procedures have similarities with the d_p.graphics application, which actually had been built as a variant of the first version of d_p.layout. In d_p.layout, instead of the pattern stamps, a series of arbitrary polygons are used as design units (DUs). The representation is still flexible to enable overlapping between DUs. However, in this case, the amount of overlapping has to be avoided as much as possible.

Likewise, the outer boundary, which was the canvas boundary in the d_p.graphics application, becomes a floor plan boundary in the layout case, and it is used to keep DUs close together, which is also controlled through formal objectives. The most important difference between the two representation schemes (graphics and layout) is the outer border of the DUs in the layout application. Each DU within the d_p.layout application has both an inner and an outer boundary. The inner boundary is used for measuring and penalizing overlap situations, while the outer boundary may be used for detecting and rewarding neighborhoods and degree of proximity.

Before the process starts, the user draws a candidate layout for each floor on an Inkscape file with the predefined layer set (see Figure 4.36). These drawings include, in minimum, north arrow, a scale line, and the outer border of each floor. Lines that indicate each facade's direction will be taken into account in evaluations, if drawn. It would also be wise to fix vertical circulations, because in its current state the system does not coordinate the placement of these on different floors. Additionally, specifying already known fixed elements such as greenery, pools, technical spaces, or entrance halls will also influence functional evaluations. Finally, for each floor, a set of fixed or movable DUs can be drawn.

In the second step, all the drawings of a building candidate are collected and parsed with a script, which scales and rotates the layouts and obtains required information. Example layouts are prepared in a similar process, whose details will be given below in the context of objectives. A set of example buildings are chosen as DU distribution targets. Using these targets in combination, the system prepares a probability mapping, which determines each DU type's probability of distribution to each of the floors. At the same time, an additional dictionary is prepared to bring together all the DUs of these target buildings with regard to their floors. For example, the DUs located in the ground floors of all target buildings are collected within a generalized ground floor. For convenience, this separate data structure will be called the 'DU distribution dictionary'.

The layouts of a building are evolved consecutively. However, the DU list is composed and distributed all at once in the beginning of the whole process. All candidate buildings share the same DU list with the same distribution to floors. As was mentioned above, there is a variety of methods for producing the DU list. First, fixed and movable polygonal DUs can be specified through the Inkscape .svg file.

Secondly, the user can submit a list of rectangular DUs by specifying their functions and dimensions. Finally, if there is remaining area in the candidate floors, the system will complete the list with reference to the DU distribution targets. Starting with the lowest one, each floor's area will be filled until a pre-specified ratio (say, %95).

The procedure for DU listing and distribution is as follows: First, fixed and movable polygonal DUs are directly added to the DU list of each floor. The total areas of these DUs are subtracted from the available areas of these floors. Secondly, the DUs given within the rectangular DU list are distributed to the remaining floors according to the probabilities given with the probability mapping. Whenever a DU is assigned to a floor, its area is subtracted from the floor area. This procedure continues until all the DUs are distributed. If all the floors are filled before the DU list runs out, the remaining DUs are evenly distributed to the floors. If, on the other hand, available floor area remains, the system uses the above-mentioned DU distribution dictionary to fill the empty floors. Therefore, if no DU list is specified by the user, the system will automatically assign a set of DUs to relevant floors.

The expectation with respect to the use of a dictionary of real DUs, instead of a probability distribution, is the ability to utilize the ratios and dimensioning of real DUs as given. However, because of both painting style and differences in the scales of buildings, a series of heuristics had to be employed. When a very large DU is drawn from the dictionary, this DU can fill or exceed a candidate floor by itself. Therefore, a limit for the maximum area of a single DU has to be specified beforehand. When such a large DU is drawn from the dictionary, the system iteratively divides this DU from its basic 'folds' as follows: There are 3 main thresholds, i.e., max. width, max. area, and max. DU area / floor area ratio. For each new DU that is drawn from the dictionary, first, these values are checked if they are below given thresholds. If the conditions are met, the DU is added to the DU list. If not, first, it is divided in two, from the length side. The conditions are checked again, and if the divided parts meet the conditions, one of them is added to the DU list. For the second one, the DU area / floor area ratio is re-checked for the sum of the two DUs. Parts of the initial DU are added to the floor until this area limit is reached or the floor is filled. This way, the dominance of the DU type is kept within a limit. The loop continues until the length value is divided by 6. If this process is not succesful in finding suitable sized DUs, the process continues with dividing the width value

from 1 to 4[22]. In this way, the width to length ratios of the DUs are more or less preserved.

Obviously, these three parameters have an important influence on the resulting plans and they have to be updated for each new scenario, which can be considered a drawback of the application. Note that the automated updating of these parameters could be possible by connecting them to the width, length, and areas of candidate buildings.

The initial placement of DUs is part of candidate initiation. Fixed DUs are initiated within their pre-specified position. There is an additional option to initiate a series of movable DUs within specific locations to be modified during evolution. The remaining DUs are initiated within the bounding box of a given floor. This bounding box is divided into a grid and each DU is randomly assigned to a grid point (Figure 4.36). The need for such grid initiation is linked with the integrated evolution of circulation elements. These elements are represented simply as another group of DUs. Therefore, to facilitate the intermingling of regular DUs and circulation elements, an optional 'striped initiation' is developed. Circulation elements are initiated in their own rows and columns, which are regularly interleaved to the other rows and columns. A candidate preparation snapshot is given in Figure 4.36. This candidate is a simplified test version of one of the used targets (Halmstad Library by Schmidt Hammer Lassen Architects). The painted candidate can be seen on the left. Note that movable DUs are later taken out of the borders of the candidate. The middle image is an initiated candidate with the painted DUs. Note that the central green area is a fixed element.

[22] The complete list of the value combinations for the folds in the applications of the thesis are: (width / 1, length / [1 – 6]), (w. / 2, len. / [1 – 6]), (w. / 3, len. / [1 – 6]), (w. / 4, len. / [1 – 6]). These values are updated in later studies as (w. / 1, len. / [1 – 3]), (w. / 2, len. / [2 – 4]), (w. / 3, len. / [3 – 5]), (w. / 4, len. / [4 – 6]).

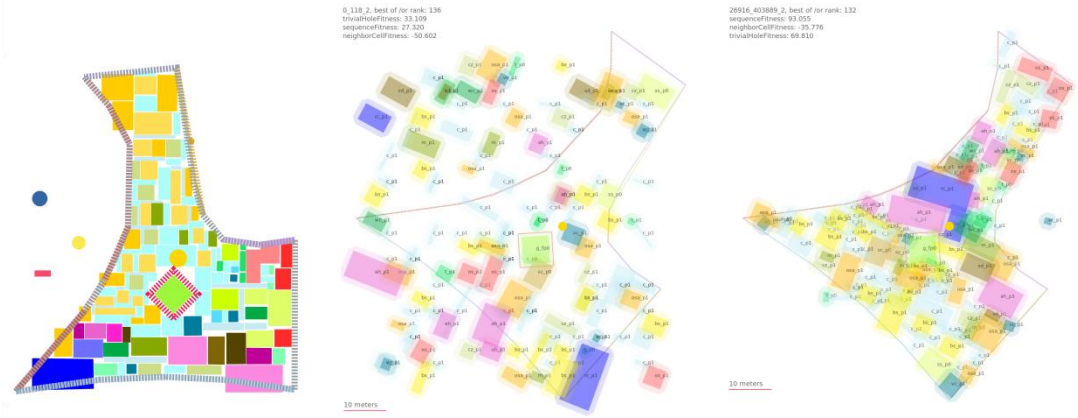


Figure 4.36 : Candidate preparation and initiation. Left: painted candidate, middle: initiated candidate, right: best result of evolution.

Another option controls the rotation of DUs according to the given floor boundary. Through linear regression, a line is fitted onto the vertices of the floor's boundary polygon, which is by default oriented towards the north. Then, each DU is rotated according to the rotation of this line. This way the DUs are initiated with a more congruent orientation with respect to the given floor's shape.

During initiation, for each objective, a series of adaptive and self-adaptive parameters are assigned to each candidate layout and to the DUs of this layout respectively.

The genotype of a candidate building comprises building-specific information together with the genotypes of the building's separate floors. In each floor's genotype, there is a single list of fixed and movable DUs. The sequence of the DUs in this list is the same for all candidate buildings. Additionally, contextual information and adaptive and self-adaptive parameters are stored at each floor's genotype. For each DU, DU type, center coordinates, inner and outer polygon coordinates, bounding box, and related geometric information are stored. Phenotype mapping information is the same for all candidates and is stored separately.

The selection and crossover operators are mostly the same with d_p.graphics. Crossover options are again one point, two point, and n point, where n point is the default choice. The procedures are the same with the d_p.graphics application. Selection operators include uniform, tournament, and fitness-proportional selection, where tournament (size 2) is mostly preferred, because systematic test sessions have shown similar selection distributions with fitness-proportional selection, but with much faster operation. The most important difference between the earlier version of

Interleaved EA and d_p.layout application is the addition of a rank-based selection operator for new population selection. The previous version was naive in the sense that it was carrying out the new population selection operation with regard only to the current objective. For the d_p.graphics application, this appeared to work, due to the independent or congruent operation of the objectives. However, during the test series of d_p.layout, it was observed that it was practically impossible to continue this naive approach and a rank-based approach is adopted.

In rank-based selection, all objectives are taken into account and the minimum rank number of an individual determines its selection probability. For each objective, each candidate has a separate rank amongst all candidates. The individuals are listed according to their minimum-ranks. Starting from the highest available minimum-rank, a desired number of individuals are selected. If a series of individuals occupy the same minimum-rank level, rank values of each individual are summed up, and the individuals are ordered in terms of the resulting values. Therefore, this approach tends to favor the candidates, which are averagely fine on all objectives simultaneously.

The swap and nudge mutations are basically the same with the d_p.graphics application (Figure 4.37). An additional neighbor swap mutation is developed which only swaps DUs if they are legitimate neighbors (outer boundaries overlap, while inner boundaries are non-violated). There are two rotate mutations; the first rotates a DU only for (±) 90 degrees. This operator can also mirror a DU, which is disabled by default. The second rotation mutation uses minor steps to rotate a DU incrementally, in order to adapt it to the orientation of neighboring borders and DUs. The last mutation operator teleports a DU onto empty spaces within floor boundaries. For this operator, empty spaces within plan boundaries are calculated and triangulated. Then a selected DU is translated to the center of a randomly selected triangle.

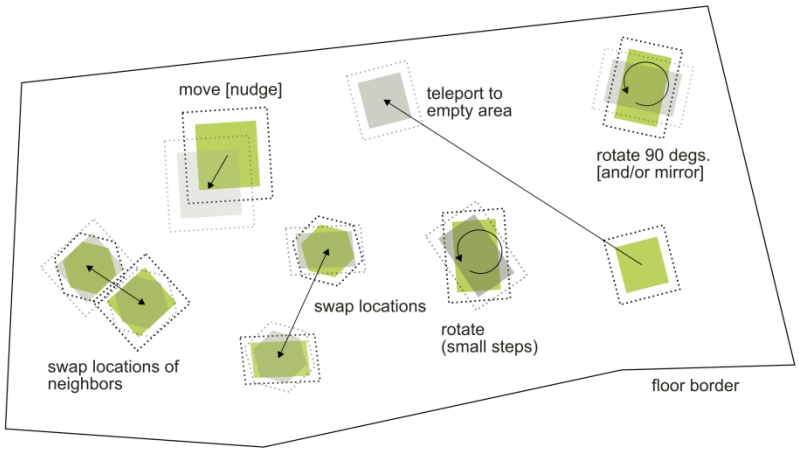


Figure 4.37 : Mutation operators.

With this multitude of available mutations, it is necessary to offer a method to decide on which one to use and how. For this reason, each objective has a mutation roulette, which includes probabilities for each mutation operator. It is possible to adaptively evolve these probabilities. During the process, for each mutation candidate, a mutation operator is probabilistically drawn from this roulette. The operators, in turn, function stochastically through a set of parameters, which is also different for each objective. The initial settings for these parameters have been determined through a long series of trials. However, some of these parameters (like the nudge step σ and swap rates) are also (self-)adaptive, and whether these would be evolved (and through which parameters) is also up to each objective. In other words, it is possible to decide on these issues for each objective separately. The most frequently used operators are the nudge and swap mutations, which influence the results most. The other operators have rather secondary roles and have minor selection probabilities. The system is based on the Interleaved EA and the evolutionary procedure is mostly the same with the d_p.graphics application, although with minor revisions and additions (Figure 4.38).

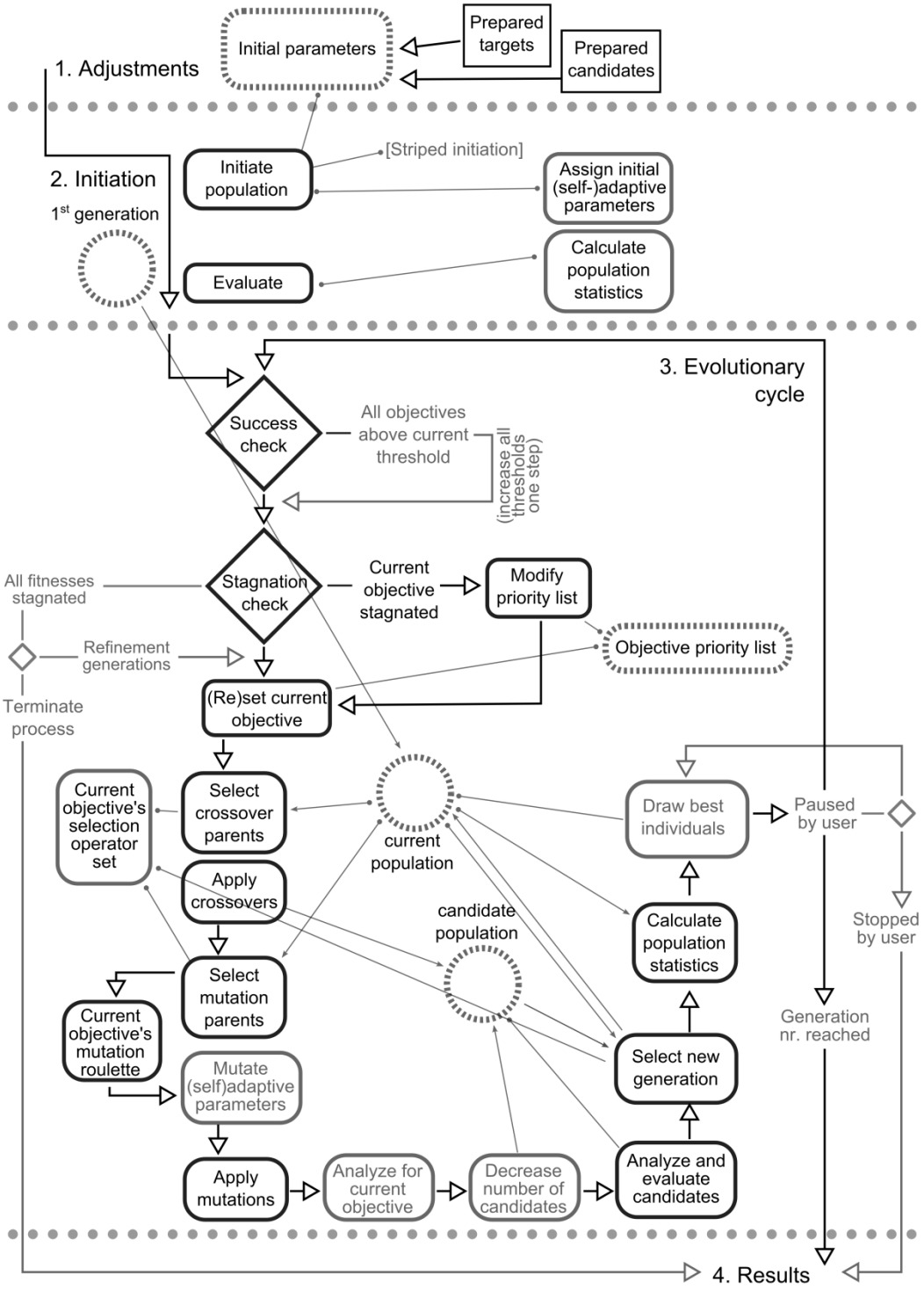


Figure 4.38 : Interleaved EA process for d_p.layout.

The most important change is the adoption of a rank-based population selection approach as described above. Additionally, objective priority list modification is revised (Figure 4.39). In cases of stagnation in the fitness improvement, if the current objective is already the last one on the list, the first option is to raise the thresholds of the other objectives. If not, the list is randomly shuffled. When the thresholds are set

to unattainable levels, this simply amounts to randomly shuffling whenever stagnation occurs, which is now the default choice. This way the problems with the priority list settings—which were quite critical—are completely eliminated.

Whenever a new individual is created within the process, it has to be evaluated for all the objectives of that evolutionary run, in order to be able to find the ranks of the candidates. However, time expenditure of each objective function varies considerably. For example, the Sequence objective consumes more time than the others. Therefore, it was practically desirable to diminish the total number of mutation numbers for such expensive objectives. In the Interleaved EA, each objective can be assigned a different mutation number, and the current objective's mutation number would be used at a specific generation. However, this was not solving the above-mentioned problem, because after each mutation, newly created individuals still had to be evaluated in terms of all objectives, including the slower ones, forcing the user to adjust to the slowest objective. Therefore, a new option is added to the system. After the specified number of mutations are carried out, newly created individuals are first evaluated only in terms of the current objective. Then their number is diminished to a pre-specified amount according to only the current objective and through the selection procedure of the current objective. Only after this reduction, the candidate population is evaluated for the remaining objectives. This procedure enables the specification of higher mutation numbers for faster objectives; hence searching through more neighboring nodes becomes possible at a specific state of the population at least for faster objectives.

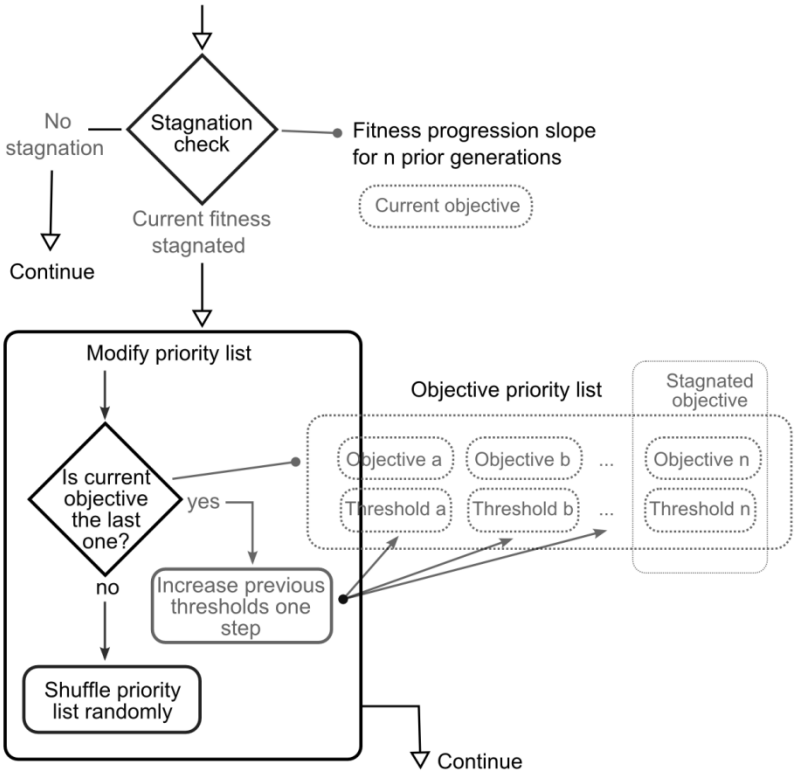


Figure 4.39 : Modification of the objective priority list for d_p.layout.

Besides optimization and programming related revisions, a series of minor practical revisions have been carried out on the Interleaved EA. First, a slope based method is substituted for stagnation control. The fitness progression graph of each objective is now fitted with a line with linear regression for n last generations (usually 100 to 300). The slope of this line shows the recent progression of the process for that objective. Thus, a stagnation threshold now denotes a minimum slope. Secondly, a very simple user interface is added for the evolutionary process. Using this interface, the user is able to pause the process to obtain intermediary process graphs or to end a floor's evolution (to yield its results and continue with the next one). Finally, a formal refinement option is added. If enabled, after the program stagnates, instead of terminating, additional generations are added only for a limited set of refinement objectives and with higher mutation and crossover numbers, which enables the searching of a higher number of neighboring nodes within the scanned fitness geography.

4.2.5 Evaluation

In total, nine objective functions are tested for the d_p.layout application. These are grouped as formal and functional objectives. There are three formal objectives,

which try to achieve the same result with different approaches, i.e., keeping DUs within borders while minimizing inner DU boundary overlap. These comprise the Out + Overlap, Proximity, and Trivial Hole objectives. There are six functional objectives. These are called Neighbor, Adjacency, Neighbor Cell, Adjacency Cell, Manual Cell, and Sequence.

As in the previous application, the aim of the Out + Overlap objective is to keep DUs within plan limits while preventing DU overlap (Figure 4.40). Because the floor boundaries are irregular in the d_p.layout application, more complicated procedures became required for calculating the penalties for the DUs that exceed the floor boundary. Otherwise, the behavior of this procedure is basically the same: give penalty to floor boundary violation and to inner DU border overlaps.

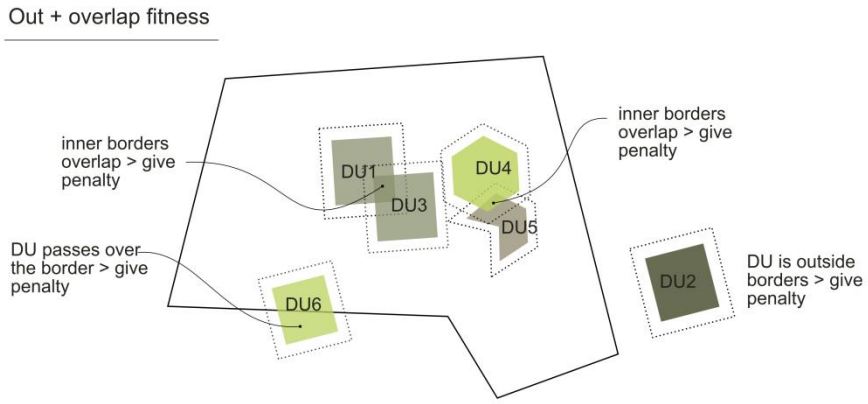


Figure 4.40 : Calculation of the Out + Overlap fitness.

An important addition in the d_p.layout evaluation procedures is the use of R-tree spatial indexing for a more efficient calculation of DU-to-DU operations, whose complexity tends to increase exponentially with the total DU number[23]. Experimentations have shown that, for the Out + Overlap objective, until around 150 DUs, the performance of the R-tree version is not significantly better. However, after this threshold the R-tree version is closer to a linear increase, contrary to the regular version, which increases exponentially; hence the improvement in the scalability of

[23] The "Rtree" module is used for spatial indexing. This is a Python wrapper for "libspatialindex" library of Marios Hadjieleftheriou (Url-18).

the system. The main burden of the R-tree approach is the requirement of updating after each mutation or crossover operation. Nevertheless, in the multi-objective case, once prepared, the same R-tree indexing is used for all objectives that can benefit from spatial indexing, thus it becomes more efficient in the multi-objective case. For example, for the Neighbor objective, the R-tree version consistently spends only around % 25 of the time spent by the regular version, for all DU numbers.

For the Proximity objective, the DUs are checked for collisions of their inner and outer borders (Figure 4.41). If two DUs collide only on their outer borders, without colliding on the inner, an award is given that is proportional with the legitimate overlap area. Thus, this objective works for keeping the DUs close and aligned. The offset value for the outer border controls the proximity of the DUs. A similar procedure checks for the DUs' neighborhood with the outer borders. To find the Proximity fitness of a candidate layout, outer boundary violation penalties are subtracted from proximity awards.

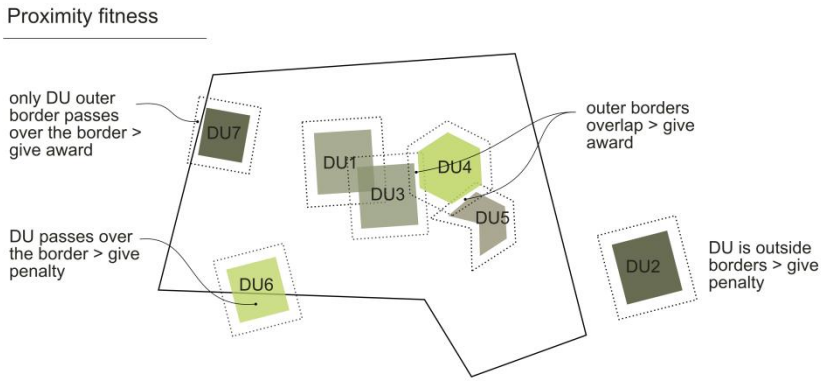


Figure 4.41 : Calculation of the Proximity fitness.

The Trivial Hole objective is much simpler (Figure 4.42). It simply calculates the ratio of the occupied area within plan borders, so that maximizing this value amounts to the minimizing of DU overlap and outer boundary violation; and implicitly, minimizing the empty areas within boundaries. However, there is no explicit mechanism for eliminating trivial holes between DUs. Thus, perhaps Proximity fitness would better be named Trivial Hole in terms of its functioning.

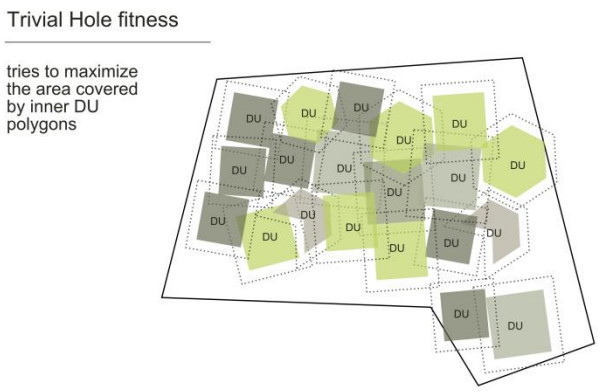


Figure 4.42 : Calculation of the Trivial Hole fitness.

The above-mentioned formal objectives depend on the inner characteristics of candidate layouts. Functional objectives, on the other hand, are measured with reference to target buildings. The Sequence objective (Figure 4.43) extracts sequences from target buildings and tries to make new plans that include those sequences. In the d_p.graphics version, the absolute positions of the DUs with regard to the canvas were being considered. However, in the plan generation example, the important information is the relative positioning of the functional units (DUs) with regard to each other. Therefore, the task is reformulated as a sequence matching procedure. As such, it is a more developed version, yet still following the methods of Smith and Chang (1999).

The first requirement for functional objectives is target preparation, which is independent from the evolutionary process. Targets are prepared from the parsed information of example buildings. The Sequence target (Figure 4.43) is prepared as follows: For each floor of an example building, rows and columns are prepared, i.e., boxes of a fixed width that are extended within the bounding box of a floor. This operation is repeated for a 45° rotated coordinate system. This way, two row-column sets will be obtained. For each of the rows and columns, all the DUs that collide with that row or column are found. Then the DUs are put in order from left to right. This yields a sorted sequence of DU types. If given, direction lines are also taken into account. Therefore, each sequence starts with a direction code and continues with DU types. Another direction code will be appended if a plan border is crossed again. This operation will continue until the end of the bounding box.

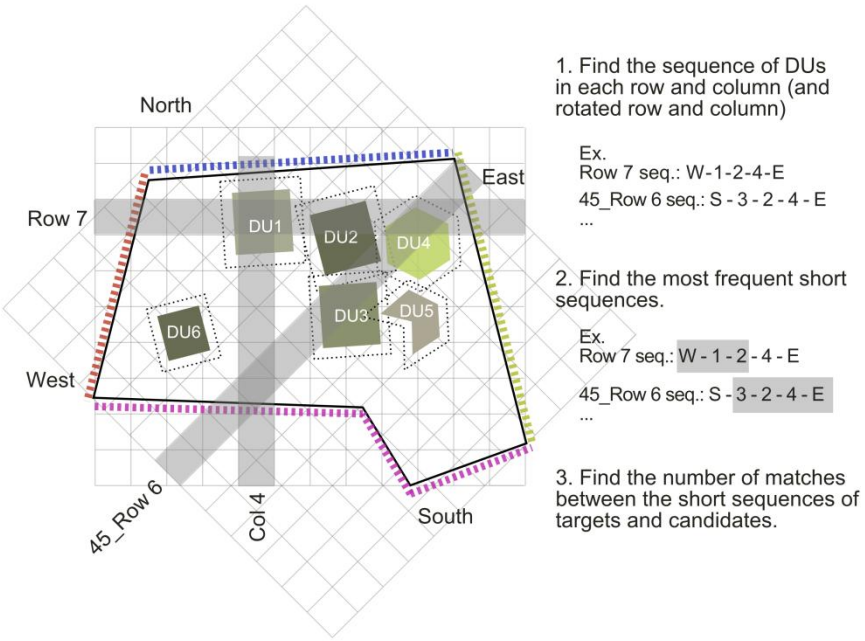


Figure 4.43 : Calculation of the Sequence fitness.

When all the rows and columns are processed this way, a series of 'long' sequences will be obtained. The width of the row or column determines the resolution of analysis. Another parameter determines a set of lengths for short sequences (by default, sequences with length 3, 4, and 5; a combination found through test series). All the long sequences are searched for matching all the short sequences of this set of lengths. Finally, the most frequent n (usually 50 to 100) short sequences and their reverses, together with found frequencies are taken as the sequence target. The length of the target influences the sensitivity of the target, but also lengthens the evaluation process during evolution. During evaluation, a candidate floor's long sequences are found in the same way, and each of the short sequences of the sequence target is sought within these. Whenever matching sub-sequences are found, a reward is given, which is proportionate with the frequency of that short target sequence.

The Neighbor objective tries to maximize the similarity of a candidate's neighborhood distribution to a target's (Figure 4.44). As such, it is thought as an alternative to adjacency matrices, which are usually static and are prepared by hand. This method enables the dynamic definition of a similar matrix through example buildings. It should also be noted that the Neighbor fitness is calculated through pattern comparison. Thus, for experimental purposes an additional Adjacency objective is developed, which is relatively similar to the traditional adjacency matrix approach.

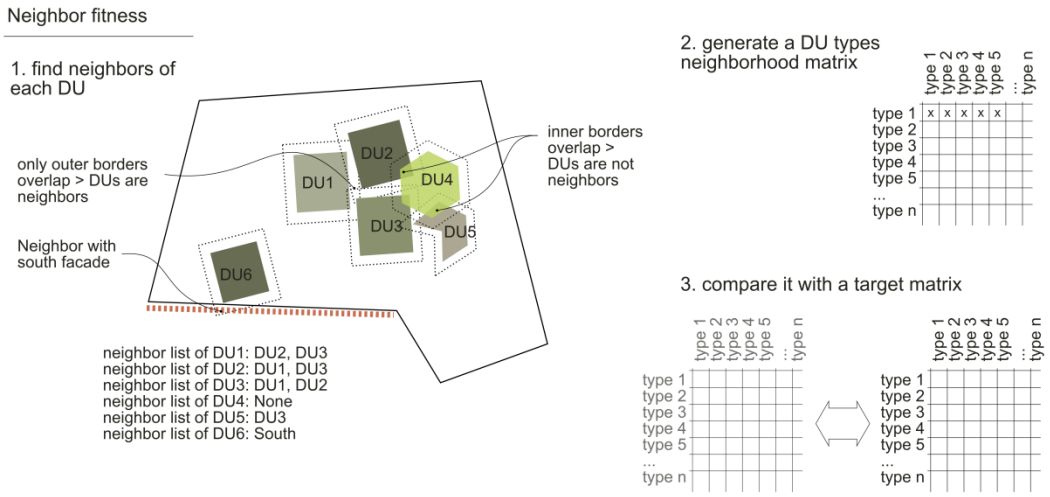


Figure 4.44 : Calculation of the Neighbor fitness.

For the Neighbor objective, a target is prepared as follows: The neighborhoods of example buildings are painted with a simple diagram. The direction that a DU is neighboring is also given with this diagram. Target preparation amounts to collecting the frequency of neighborhoods between DU types within a neighborhood matrix, from all floors of a single example building. The rows and columns of the matrix are symmetrical and their sequence is fixed for all applications. The frequencies of neighborhood types are normalized by the maximum frequency. This way the matrix becomes a neighborhood pattern, which can be compared with other buildings.

For candidate evaluation, neighborhoods of DUs are extracted from legitimate overlaps, i.e., overlap of outer boundaries while inner boundaries remain non-violated. Collision with the direction lines are used for the detection of direction neighborhoods. When the neighbors for each DU type are determined, this information is again converted into a normalized neighborhood pattern matrix. For fitness calculation, the absolute values of the differences of corresponding target and candidate matrix cells are summed up. The resulting value is negated to be able to treat this error value as positive fitness. For the Adjacency objective version, the frequency values in the target matrix are taken as adjacency rewards; the most frequent (1) being the most desired adjacency. When a cell value is subtracted from 1, this gives a penalty value; the most desired adjacency gets a penalty value of zero, the least desired adjacency gets the maximum penalty, that is, 1. To find the fitness value of the layout, for each target cell, its penalty value is multiplied with the frequency of the corresponding cell of the candidate. The resulting errors are

summed up and negated for finding the positive fitness. By default, the Neighbor objective is preferred over the Adjacency version, as it has the advantage of more fine-grained similarity comparison.

Finally, there is a group of grid cell objectives. Similar to the Neighbor objective, the Neighbor Cell objective (Figure 4.45) tries to evolve plans that are similar to a target neighbor matrix distribution, according to DU type. However, in this case the analysis method for the spatial adjacencies is different. In addition, adjacencies to directions are not taken into account for the Cell type objectives. When used together the two objective types enable a more fine grained comparison with a target library. Also separate targets may be used for each objective. For fitness calculation, a fixed sized square query cell is placed over each DU's center. Types of all the DUs that collide with a DU's query cell are added to the cell neighbor list of that DU. The query cell is square instead of circular, because this way the overlaps can be directly determined through R-tree indexing, which makes the analysis procedure very efficient, albeit a bit coarser; because in this approach DU collisions are calculated with regard to DUs' bounding boxes.

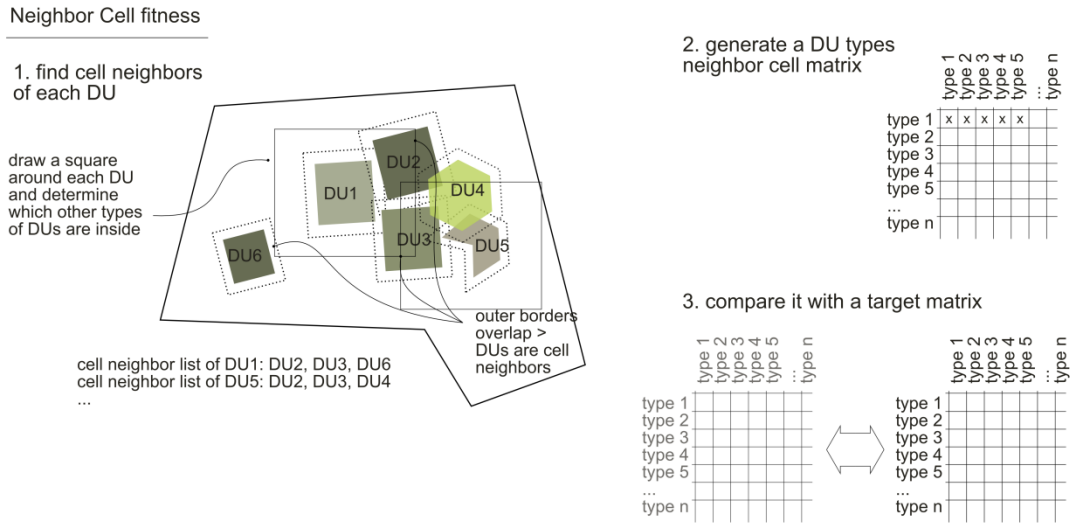


Figure 4.45 : Calculation of the Neighbor Cell fitness.

For candidate evaluation, the target's matrix is compared with the candidate's. As with the Neighbor objective, Neighbor Cell objective calculates and negates the difference between two matrix patterns while the Adjacency Cell version calculates the negative of the total penalty.

Finally, the Manual Cell objective (Figure 4.46) tries to bring several DU types together by looking for cells that conform to manually given cell lists. This objective is developed for obtaining already known functional groupings, like building cores, which frequently bring toilets, elevators, staircases, and technical elements together. To achieve its aim, the procedure regularly divides the bounding box of a candidate layout and traverses through each grid point with a square query cell for sampling the existing groupings. Each sample is compared with given desired cell groupings. The aim is to find all the DU types close together as overlapping a query cell. Each query cell returns a ratio within range [0-1] that reveals its success. A ratio of "1" denotes that all desired DU types are found together within a cell. Whenever a value of "1" is found, the procedure ends. Otherwise, it keeps searching through the whole grid and returns the best-found ratio. If there are more than one desired cell lists, the average value of all the best-found ratios is returned.

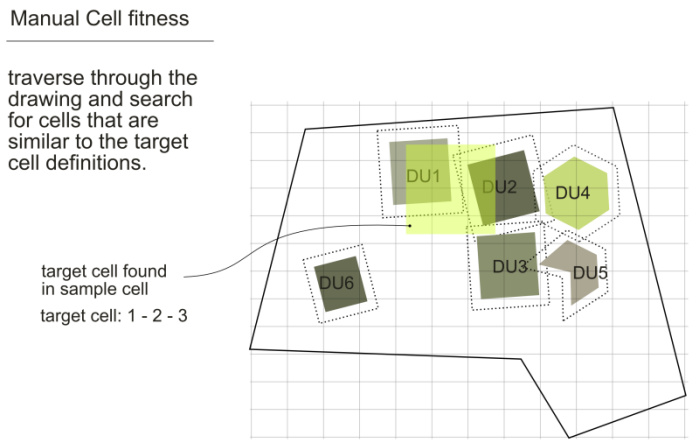


Figure 4.46 : Calculation of the Manual Cell fitness.

4.2.6 Test series and verification

In this section, a series of tests will be presented to verify the functioning of the d_p.layout system. The verification process has unfolded as an open-ended, creative enterprise, because of the combinatorial nature of the operation of different variables and the complex nature of the resulting layouts. In other words, the following tests had to be tailor-made for the d_p.layout system and their effectivity in determining the success of the system will remain a matter of interpretation and debate.

The overall question to be answered is, "does the system carry out its tasks as intended and to which degree?" The question involves both technical aspects and real

world usage. This section will rather concern technical issues, while questions about the effectivity of the interface, adaptability of the system to different contexts, and overall practical usage are left to the next chapter.

Several questions regarding the functioning of the Interleaved EA has already been addressed within the context of d_p.graphics application. However, new questions arise in the d_p.layout application, because of the new objective functions, rank-based multi-objective evolution, and the proliferation of adaptive and self-adaptive parameters. Because these questions are not separable from the regular functioning of the system itself, they will be examined within multi-objective test series. The two most important aspects that have to be re-examined in this new application are the effectivity of parameter separation per objective and adaptive parameter control.

Two sets of technical questions emerge in this context: Firstly, with regard to objective functions, "is each objective able to drag a candidate layout population towards the desired situation?" "Does each objective work reliably for a range of dissimilar cases?" "Does each objective work within practically acceptable time limits?" Secondly, "how do the objectives operate in a multi-objective setting?" "Is the system, in its totality, able to function as desired?" "Is it reliable and fast?" "What is the effect of the number of objectives?" "What is the effect of the number of DUs (Is the system scalable)?" "Is Interleaved EA more advantageous than a regular rank-based evolution approach?" And, "how do the different adaptive and self-adaptive schemes compare with the non-adaptive cases?"

To answer the first set of questions, a special test scenario has been developed for each objective. For each of these tests, first, evolutionary progression is examined by means of fitness graphs. The curve of these graphs is an indicator of the balance between population number and variation operators. It is also used for examining the system's reliability and regularity through distinct applications. Secondly, the results are visually inspected to see if the desired results are being obtained. Therefore, the tests had to be designed to facilitate easy visual interpretation, which is achieved by simplified candidate and target definitions.

For this aim, first, the formal objectives will be assessed through a set of single objective tests. Secondly, for the functional objectives, two-objective test series will be illustrated. These tests have to involve at least two objectives because of the

dependence of the functional objectives on the formal ones due to permissive problem representation. Indeed, all tests implicitly assess the formal objectives, because there is always at least one formal objective at work.

The second set of questions targets multi-objective cases, for which, systematic experimentations have been carried out for seven different system scenarios. 100 tests have been carried out for each scenario with 50 DUs and 6 objectives. The average fitness graphs of the progression of mean fitnesses of these 100 tests will be superposed for each scenario according to different objectives.

To test for the comparative effect of the number of DUs, which is very important in assessing the scalability and limits of the system, hence its adaptability to new situations, formal objective test series have been carried out for 50, 150, and 250 DUs. Indeed, most experiments implicitly assess this issue and the tests for functional objectives can also be seen to test for simpler scenarios with smaller DU numbers. In addition, a series of tests have been carried out for 3, 6, and 9 objectives and for 50, 150, and 250 DUs, generating a matrix of 9 conditions. However, due to time and performance limitations, these tests could not be carried out in a systematic fashion (i.e., around 5-10 tests for some cases and just one or two for the most complicated 250 DU cases). It should be noted that, all of the tests have been carried out on a regular quad-core desktop computer. This limitation was somewhat useful, because the system would eventually be used on such computers, and understanding the relative speed of different settings was important.

All tests have been carried out over single floor buildings. In a multi-storey scenario, the floors are evolved consecutively, i.e., independently. Therefore, the only property that should be tested on multiple floors is DU distribution. This functionality will be exemplified and evaluated later in this chapter, in the context of workshop applications.

4.2.7 One and two objective test series

The illustration of the verification tests will start with the formal objectives. During the initial parameter finding process, the Out + Overlap, Proximity, and Trivial Hole objectives have been tested both separately and together. The Proximity objective had been developed for clinging DUs close together. However, initial tests have shown that its success is limited for this task; and it is slightly worse than Out +

Overlap in preventing DU overlaps. Yet it does not negatively interfere with the other formal objectives. Therefore, the Proximity objective is separated for experimental purposes (for the nine-objective series). A series of additional tests have been carried out to evaluate the joint performance of the Out + Overlap and Trivial Hole, which have shown complete accord. In single objective runs, the Out + Overlap fitness progresses faster, however, in complicated cases it tends to leave a few DUs outside the floor boundary. Although both are consistent and sufficiently reliable for all scenarios, the Trivial Hole objective is more reliable and stable and fitness graphs of the Trivial Hole fitness are easier to examine, compared to the Out + Overlap, whose fitness range is generally too wide and progression is very steep.

Below, single objective test series for the Trivial Hole objective will be illustrated for 50, 150, and 250 DUs. Each test has been carried out at least three (for the 250 DU case) or four (50 and 150 DU cases) times to observe potential differences in individual runs. It will be possible to examine results and example fitness graphs for the Out + Overlap objective in the context of subsequent tests.

Because it is easier to show that they work, the tests for formal objectives have been carried out through a real-like scenario, based on Halmstad Library, whose plan border is quite complicated with a hole inside. The 150 DU version is obtained by simplifying the target by painting over the original. The DUs are mostly kept in their places but their shapes are simplified. The 50 and 250 DU alternatives have been generated from this first.

In Figure 4.47, the top row gives the initiation sequence for the 50 DU version. The middle row illustrates example results of four evolutionary runs with various generation counts. The graph on the bottom row consists of the superposition of the mean and worst fitness graphs of these four example runs. The black line represents the run that was set to exactly 10000 generations. The others were set to exit whenever the process stagnates (i.e., converges). For the aims of d_p.layout, inspecting the mean and worst fitnesses is more important than the best-so-far fitness. It can be seen on this graph that the Trivial Hole objective results in a stable and reliable process; it is almost the same for all trials. The range between mean and worst fitnesses is rather narrow; the mean result is close to the best result (given on the middle row) and even the worst result is satisfactory. This situation recurs for the other DU numbers as will be seen. When inspected visually (outer DU borders are

made transparent for easier evaluation), it can be seen that the DUs are located and rotated to conform to each other's inner borders as well as plan borders.

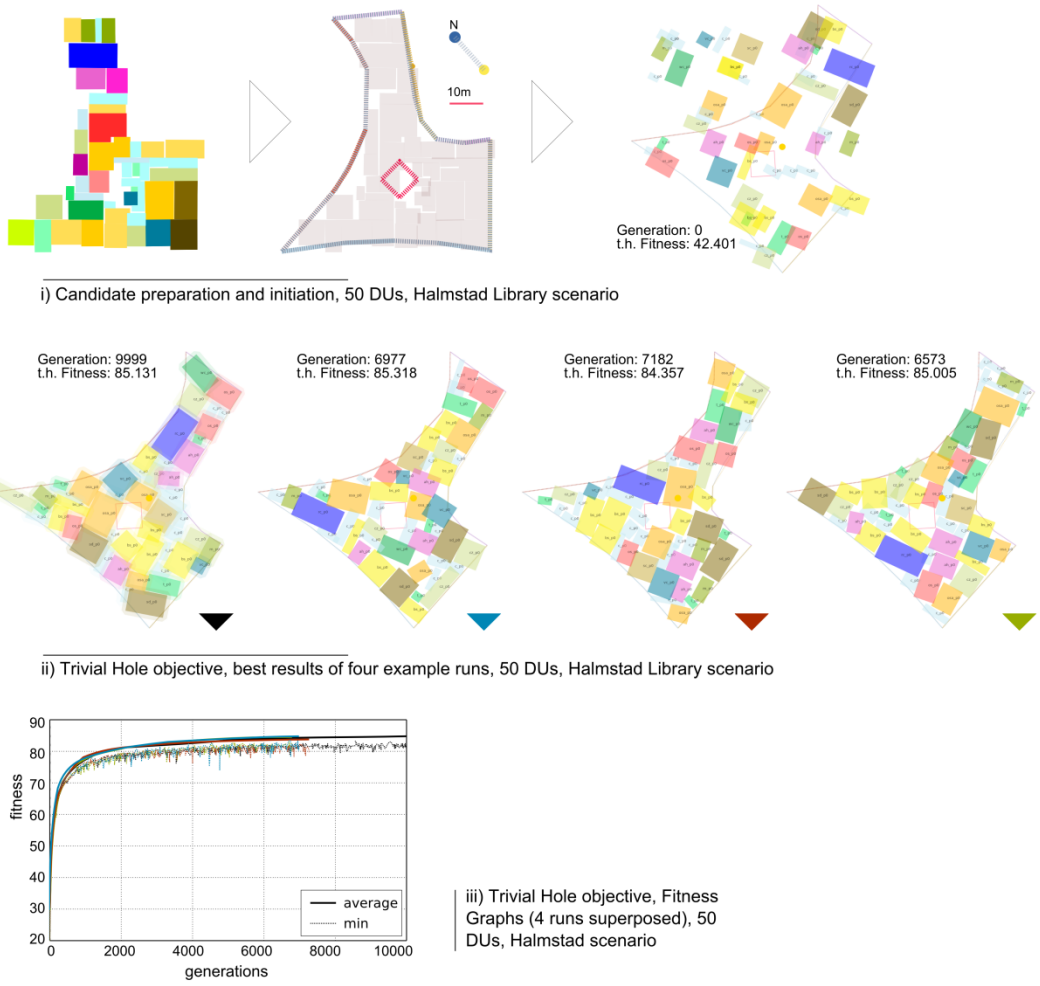


Figure 4.47 : Trivial Hole objective, for 50 DUs.

Figure 4.48 illustrates the 150 DU scenario. The results are mostly similar with the 50 DU case. Even with 150 DUs, the system is able to evenly distribute the DUs within plan borders; however, more empty spaces are left in between DUs in this case. Outer DU borders are also drawn (half transparent) in the leftmost example. The real separation lines between DUs can be thought to lay in between inner and outer borders, hence these have to be found through interpretation. The same flexibility and permissiveness that make the system adaptable and scalable are the causes of this vagueness. Nevertheless, this is congruent with the proposed usage of the system as will be discussed through the workshop applications in the next chapter. The example fitness graph is similar with the previous ones, yet appears to converge faster due to the increased complexity of the problem.

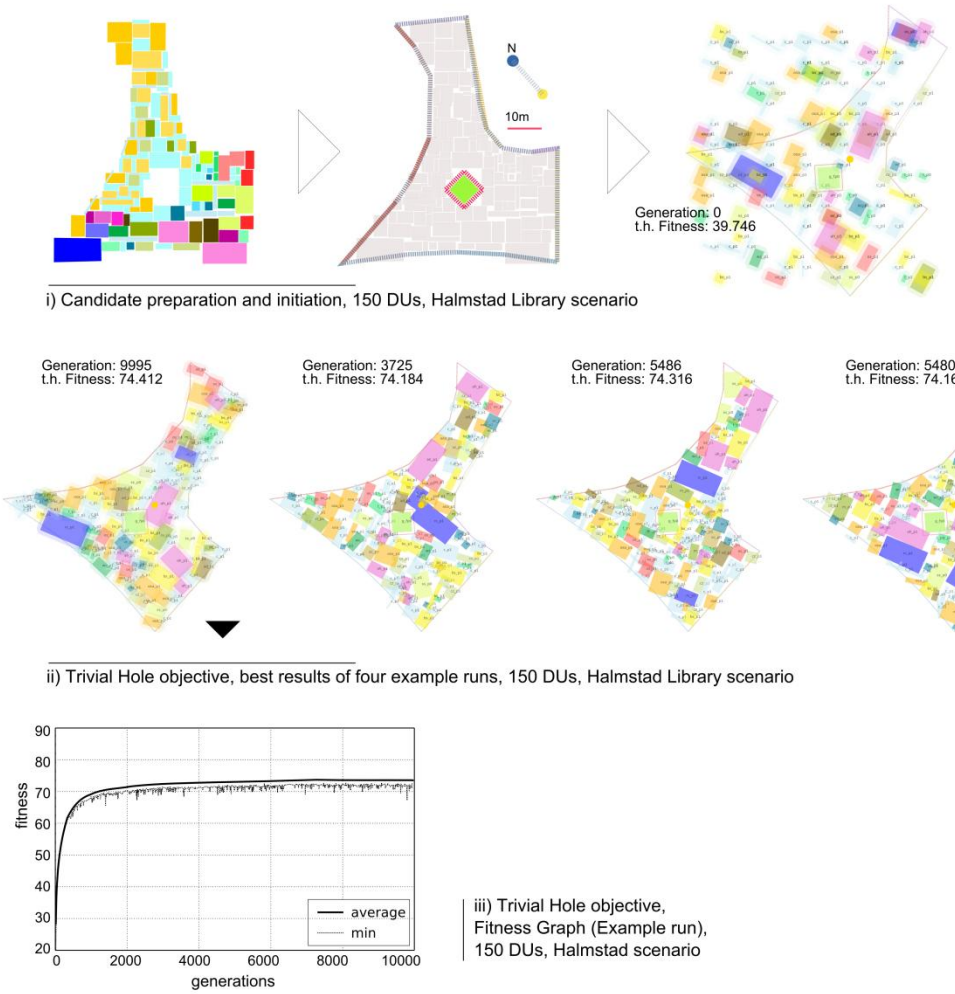


Figure 4.48 : Trivial Hole objective, for 150 DUs.

Figure 4.49 illustrates the 250 DU case. Due to the complexity of the problem 20000 generations have been tried in this case; however, the consequence of the increased complexity is early convergence, which renders the increase in generation count useless. It should be noted that the population and mutation numbers have been kept the same for all these tests (Population: 150, mutation: 15, crossover: 0), while an increased complexity of a problem could have been countered with an increase in these parameters. However, first, an increase in mutation numbers has a performance cost. Secondly, an obligation to manually adjust the parameters for each new scenario is not much desired. Nevertheless, simple heuristics may be employed to automatically alter these parameters with regard to DU numbers while enforcing upper limits.

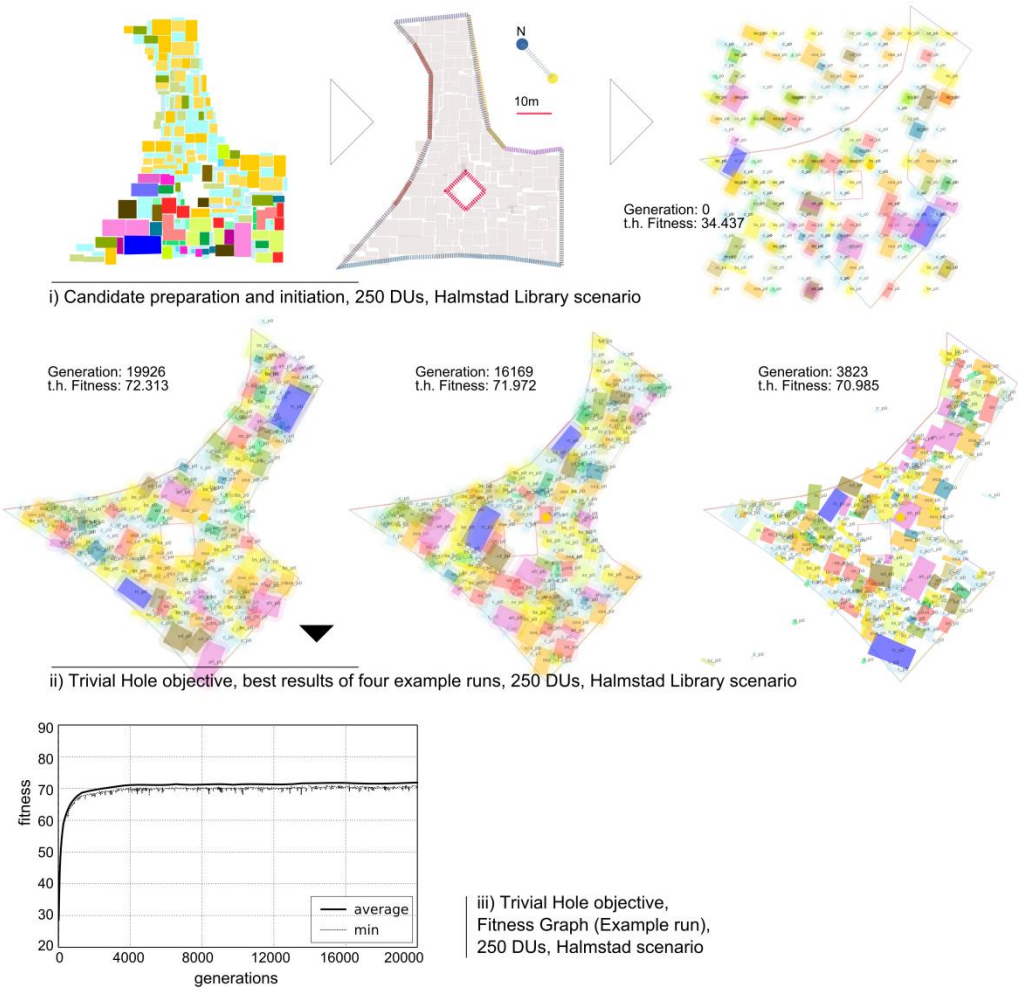


Figure 4.49 : Trivial Hole objective, for 250 DUs.

Visual inspection of the results in Figure 4.49 shows that the amount of unwanted overlaps have increased in this scenario (the rightmost example gives only inner borders), which is not surprising. However, the main problem for such a high DU number is rather a potential problem with overall organization. Indeed, even the most complicated example layouts did not exceed a number of 150 DUs throughout our experimentations. And apparently, handling such a complicated plan in only one stage is not a viable method. If such large and complicated plans are first separated into higher regions corresponding to main functional groupings, then these could be handled separately, which would also result in more manageable problems; hence a hierarchical approach appears more appropriate for these situations. Still, although not perfect, it appears that the system is able to function for the Trivial Hole objective even with such a high DU number.

The situation is more complicated for the functional objectives. The main requirement is to produce target-candidate pairs that would reveal whether an objective guides an evolution in the desired direction or not. As has been stated above, each functional objective is paired with a formal one for these tests. The most complicated verification problem is raised by the Sequence objective. The basic question is, "will the sequences given within the target layout reappear in the candidates?" The fitness function is so designed that the sequences do not require an absolute matching of the target and candidate layout arrangements. It just tries to maximize the most frequent sequence types in the candidate. The tests will start with the simplest case, which comes close to searching for an absolute similarity of the target and candidate. Then through a more complicated case, the real functioning of the objective will be examined.

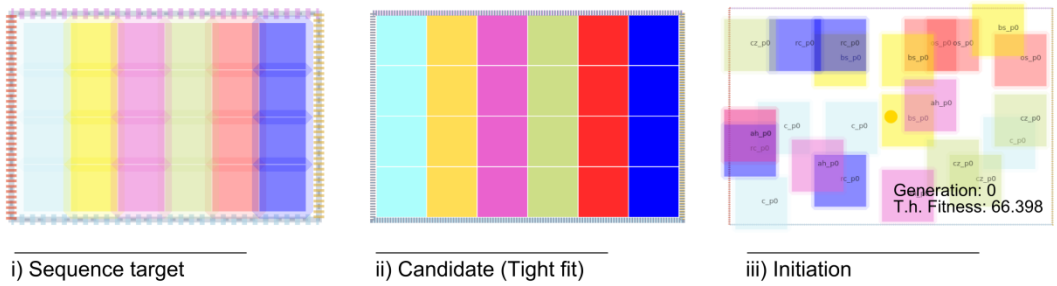


Figure 4.50 : Target and candidate (tight fit) for Sequence objective.

Target, candidate, and initiation for the first test series can be seen in Figure 4.50. The candidate is almost the same with the target. In Figure 4.51, results of a series of control tests are given. These examples are evolved only with the Trivial Hole objective, thus their functional arrangements are totally random. These will enable us to assess if the test results could be obtained randomly. For the sake of simplicity, rotation mutations have been disabled for these series.

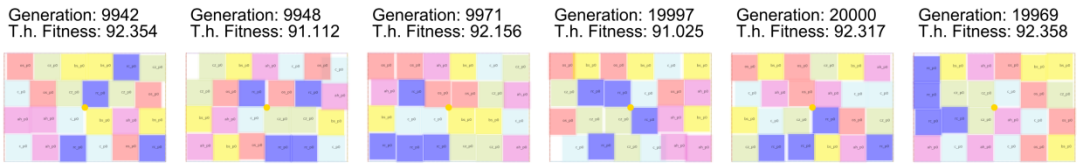


Figure 4.51 : Control evolution with only Trivial Hole objective (tight fit scenario).

The best results of two test series (population: 150, mutation: 15, crossover: 0) are given in Figure 4.52. The first row concerns the simplest case, which involves short sequences of length 4 and 5 that have been searched only within rows and columns

of the candidates (without 45° rotation). The tests in the second row take both 0° and 45° rotated row and columns into account. Each image is the best Sequence objective result of a single evolutionary run. On the left, 10000 generation tests are given, while 20000 generation tests are given on the right side.

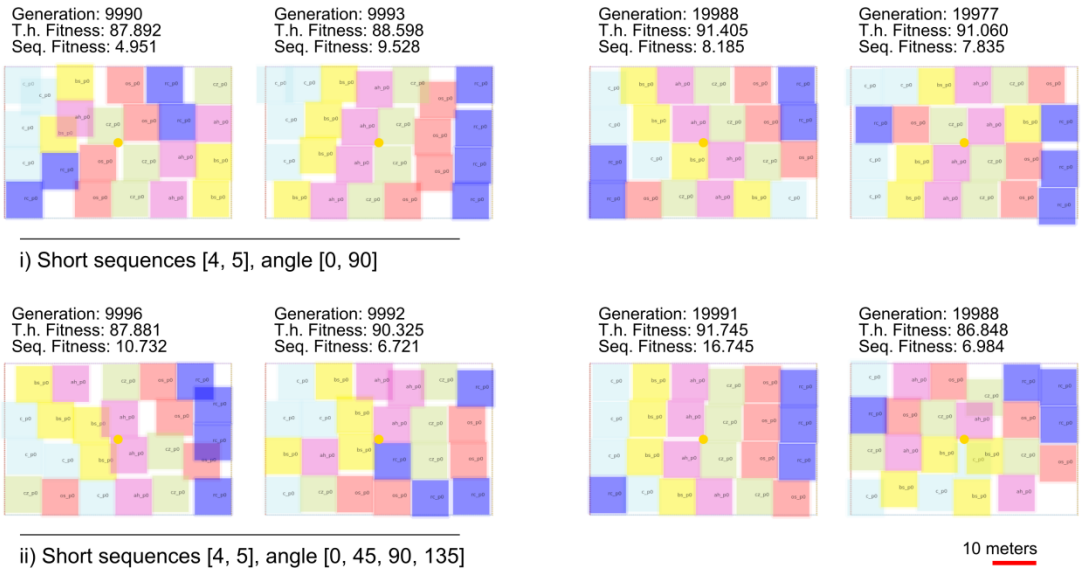


Figure 4.52 : Sequence objective verification – Sequence + Trivial Hole, tight fit candidate, best results of 0° and 0° + 45° versions – Sequence grid width: 2 meters.

Visual examination of the simplest case clearly reveals that the Sequence fitness is able to indicate the relative similarity of a target and a candidate. The visually fitter individuals can be seen to be also better in terms of their Sequence fitness values. The effect of the inclusion of direction lines is also obvious; the obtained results tend to follow the exact sequences, not the mirrored or rotated alternatives, which should have been obtained in equal numbers if the directions were not being considered. Because the system shares its time amongst a series of objectives, when a population finds a more expedient path towards a better fitness region for its functional objectives, it also becomes able to allocate more time to the formal objectives. It can also be noticed that with such a simple and strict problem definition, fitness increments to be climbed by the system are too large. This results in inconsistent fitness progression. The inconsistency of the process is also observed in the fitness graphs. Example fitness graphs are given in Figure 4.53. The spikes on the graphs indicate rapid oscillation between the two objectives, which is the expected behavior of the Interleaved EA. However, the progression of the Sequence fitness is rather

peculiar. As can be seen with the results, although near-perfect results were possible, much worse results have also been obtained with the same generation count.

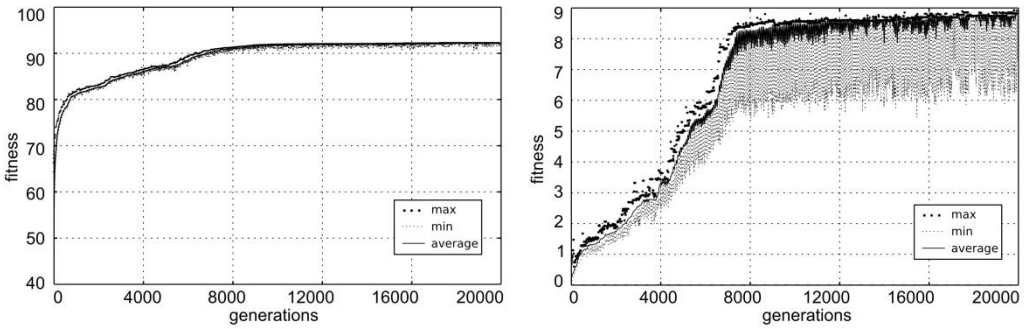


Figure 4.53 : Sequence + Trivial Hole, example fitness graphs.

A second series of similar tests with the Out + Overlap objective can be seen in Figure 4.54. With slight differences in the problem setting, the results are similar. Example fitness graphs are given in Figure 4.55.

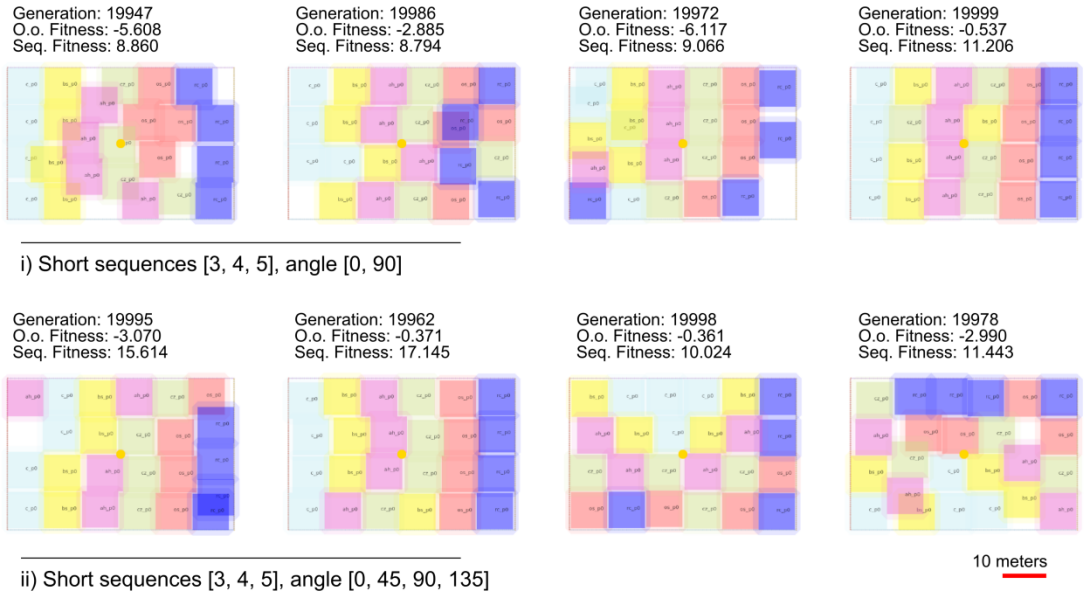


Figure 4.54 : Sequence objective verification – Sequence and Out + Overlap, loose fit candidate, best results of 0° and 0° + 45° versions – Sequence grid width: 2 m.

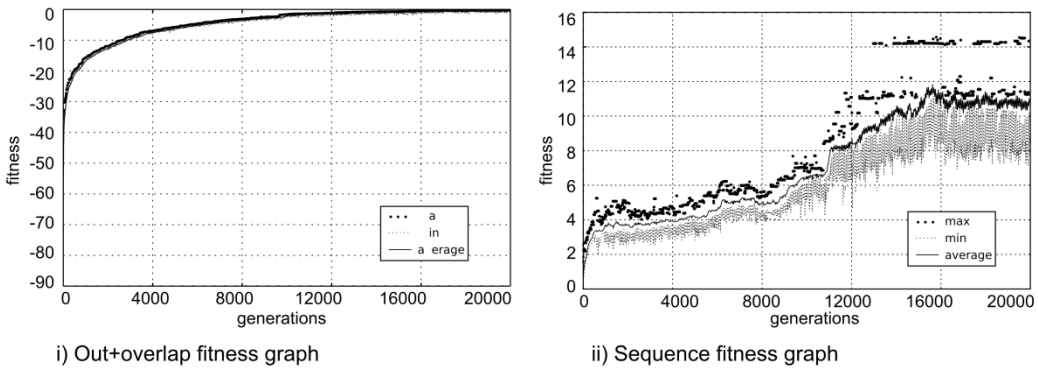


Figure 4.55 : Sequence and Out + Overlap, example fitness graphs.

The close similarity pursued in the above trials is indeed not the desired behavior for the Sequence objective. Therefore, a second test series has been devised with 2x2 times the amount of DUs (i.e., 96 DUs) (see Figure 4.56 for initiation and Figure 4.57 for random control examples). Although a bit more difficult in legibility, these series will better assess the behavior of this objective. The short sequence length set is [3, 4, 5] and rotations are 0° and 0° + 45°.

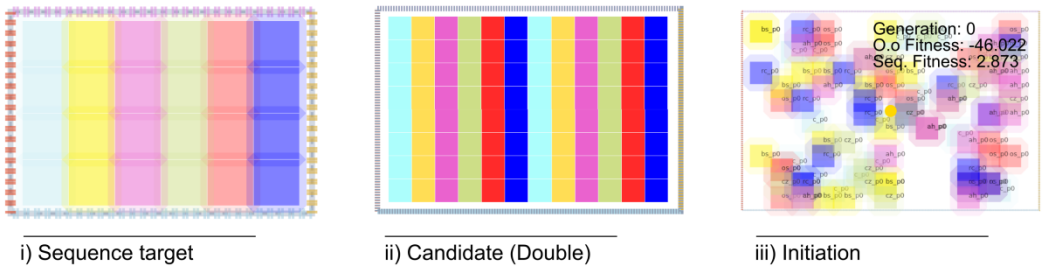


Figure 4.56 : Target, candidate, and initiation for Sequence objective.

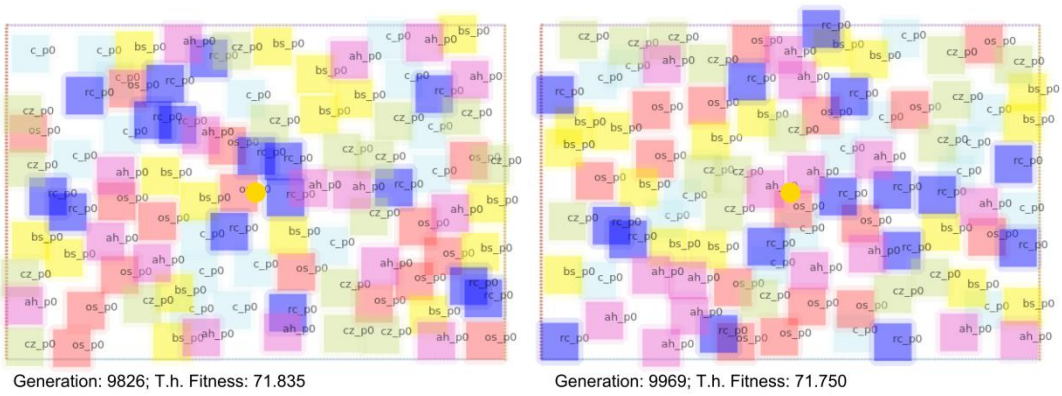


Figure 4.57 : Random control runs with only Trivial Hole objective.

Example results are given in Figure 4.58. On the upper row, the 0° tests are placed, while 0° + 45° tests are given below. Figure 4.59 illustrates example fitness graphs from these tests. It can be seen in the fitness graphs that both fitnesses were still

improving regularly at the point of termination. Although the Out + Overlap fitness' improvement speed was decreasing, it appears that the Sequence fitness would continue improving. In general, with a more flexible problem setting, smaller fitness steps, and more resources to exploit, more regular and reliable evolutionary processes have been obtained. When compared with the control examples in Figure 4.57, it can be observed that both the vertical same-color sequences and the horizontal color sequences were being generated in both 0° and 0° + 45° trials. Obviously, the 0° + 45° version has greater resources for exploitation, which results in conspicuous regular formations and higher fitness values. It can also be seen by visual inspection that the individuals that display more of the desired sequences obtain higher Sequence fitnesses.

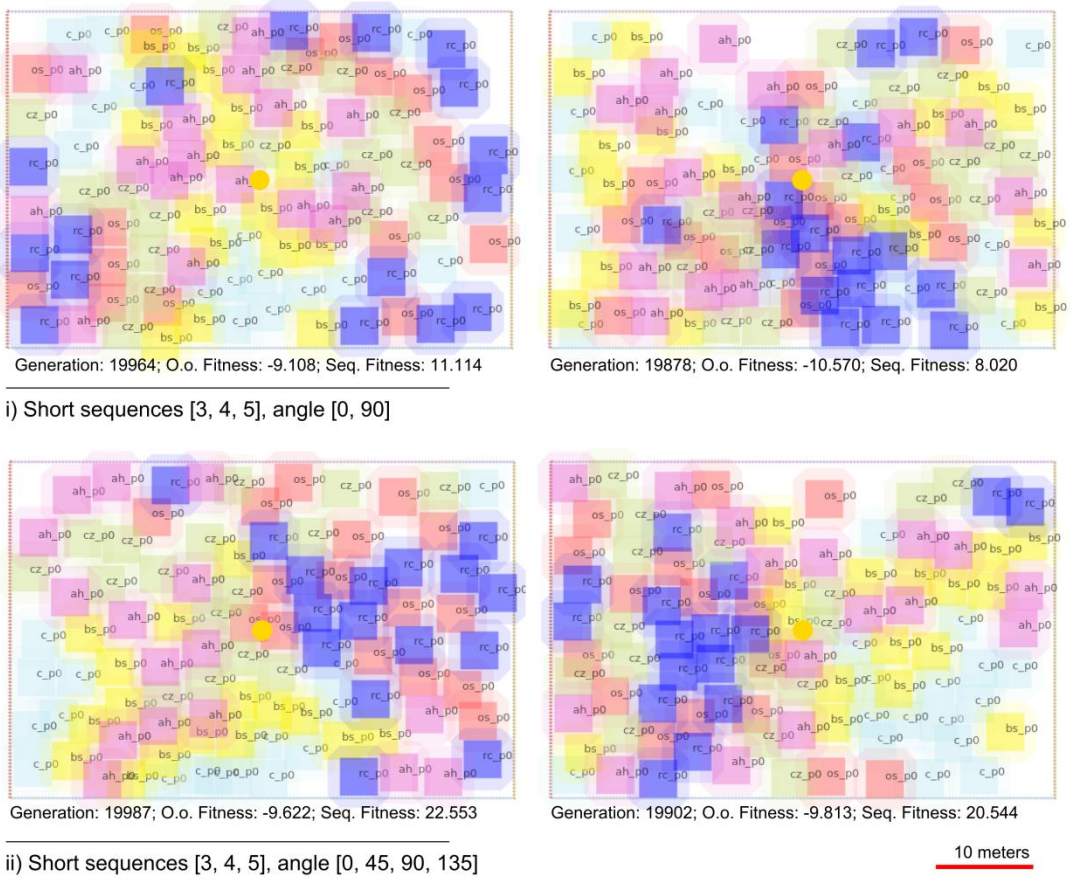
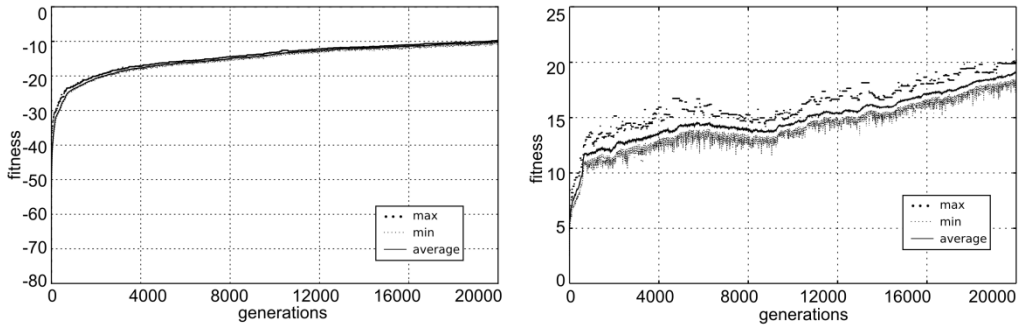


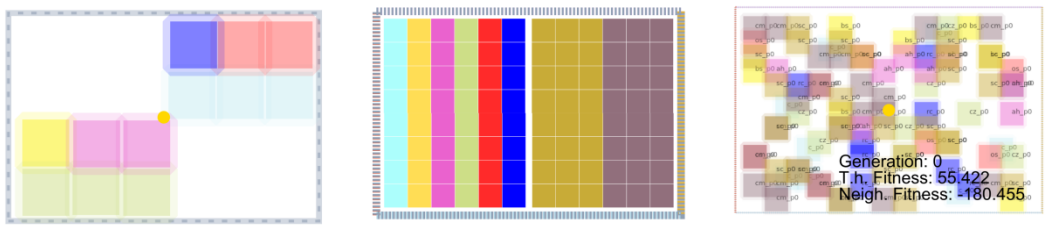
Figure 4.58 : Sequence objective verification – Sequence and Out + Overlap – Sequence grid width: 2 m.



i) Out+overlap fitness graph ii) Sequence fitness graph

Figure 4.59 : Sequence and Out + Overlap, example fitness graphs – sequences [3, 4, 5], rotations [0°, 45°].

The goal of the Sequence objective is to obtain a set of interrelated sequences. For the Neighbor and Adjacency objectives, the goal is to bring and keep together several types of DUs, with respect to target neighborhood patterns. The target for these objectives (Figure 4.60) consists of two almost distinct groupings, which are expected to reappear in the results. Half of the DUs in the candidate belong to the types in the target. The DU types of the other half have no target information. These are given with two colors (i.e., types) to see how the uninformed DUs will be distributed within the resulting layouts.



i) Neighbor and Adjacency target ii) Candidate iii) Initiation

Figure 4.60 : Neighbor and Adjacency objectives; target, candidate, and initiation.

Example best results are given in Figure 4.61 (population: 150, mutation: 15, crossover: 0, rotation mutations are enabled). The left column gives the trials with directions enabled. Both objectives are successful in bringing relevant DUs within distinct groupings. The effect of directions appear negligible, which is understandable because the award for direction and DU neighborhood is the same and the system appears to exploit the more prolific DU-to-DU neighborhoods for this scenario.

Figure 4.61 : Neighbor and Adjacency objectives, example best results – DU offset: 1.2 m.

The fitness graphs in Figure 4.62 and 4.63 indicate that the processes had not yet converged in 20000 generations. However, in practice, there will rarely be a longer period for an objective and these rather intermediate results are sufficient for the evaluation of the tendencies of these fitness functions. The main difference between the two objective functions is that the Adjacency fitness can be maximized with over-exploiting the most advantageous DU relationships; light blue and light green in this scenario. However, the Neighbor objective penalizes a disruption of the target pattern, hence it becomes difficult to lean too much on the exploitation of any relationship on its own; the overall neighborhood pattern has to be conserved. This can be evaluated by the inspection of yellow and dark blue cells. Because their overall fitness gain is lower, the Adjacency objective tends to omit improving the neighborhood patterns of these cell types and concentrates on the more frequent DU types, so that we can observe free yellow and dark blue cells with no desired neighborhood at all. On the other hand, as expected, the Neighbor objective appears

more consistent in improving with all types of DUs and mostly succeeds in regenerating the target neighborhood patterns.

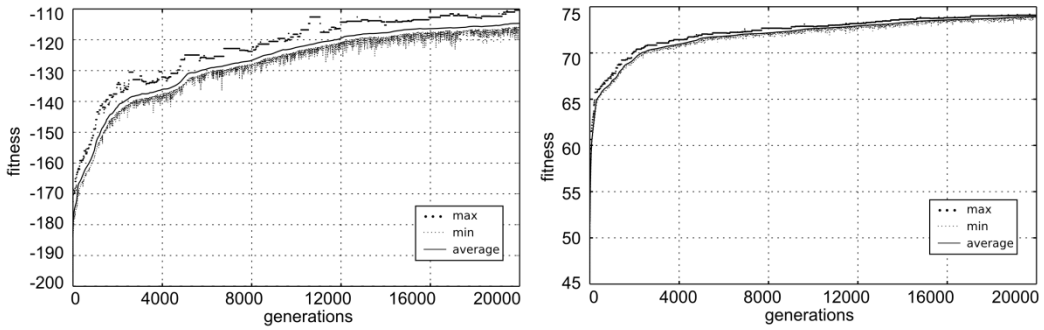


Figure 4.62 : Neighbor + Trivial Hole, example fitness graphs.

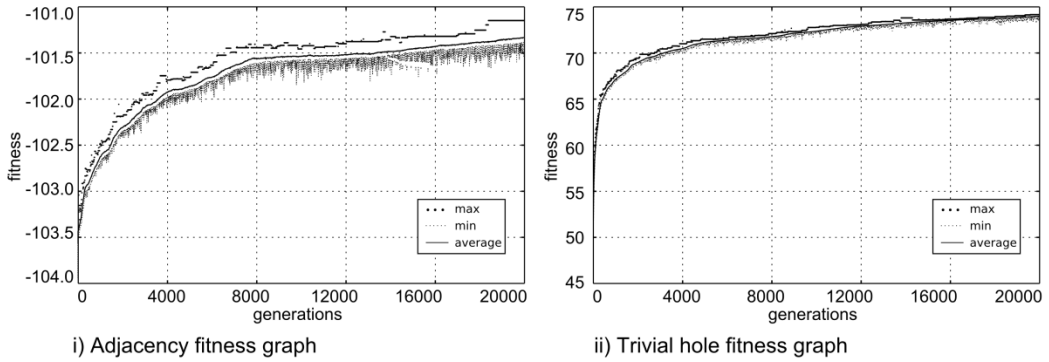


Figure 4.63 : Adjacency + Trivial Hole, example fitness graphs.

A similar distinction applies to the Neighbor Cell and its Adjacency Cell version. The targets and candidates for these objectives are given in Figure 4.64. In this setting, for the Neighbor Cell objective, a dark blue DU would like to be near two light blue DUs and a red DU. Likewise, a light blue DU would prefer the neighborhood of one light blue, one red, and one dark blue DUs. In the Adjacency Cell version, because both dark blue and red DUs award light blue neighbors more than the others, the light blue DUs would tend to get collected near each other. However, the same cause would affect an intermingling of the dark blue and red DUs with the light blue ones, and the situation would get balanced to an extent. As a result, combinatorial effects of the two versions appear similar as can be seen in Figure 4.65. The situation could have been different if a target like the one used for the Neighbor objective had been used. In such a case, the advantage gap between the most and the least advantageous proximities could have made the least advantageous to be omitted in the Adjacency Cell version. This shows that the combinatorial effect

of these objectives is strongly dependent on the scenario. Example fitness graphs of the trials are given in Figures 4.66 and 4.67.

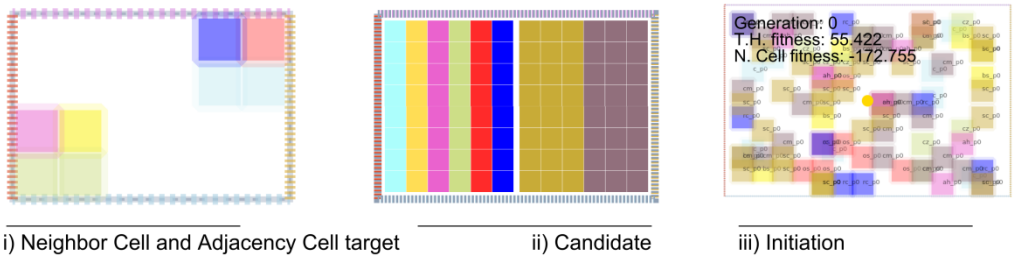


Figure 4.64 : Neighbor Cell and Adjacency Cell objectives; target, candidate, and initiation.

Figure 4.65 : Neighbor Cell and Adjacency Cell objectives, example best results – Query cell width: 8 m.

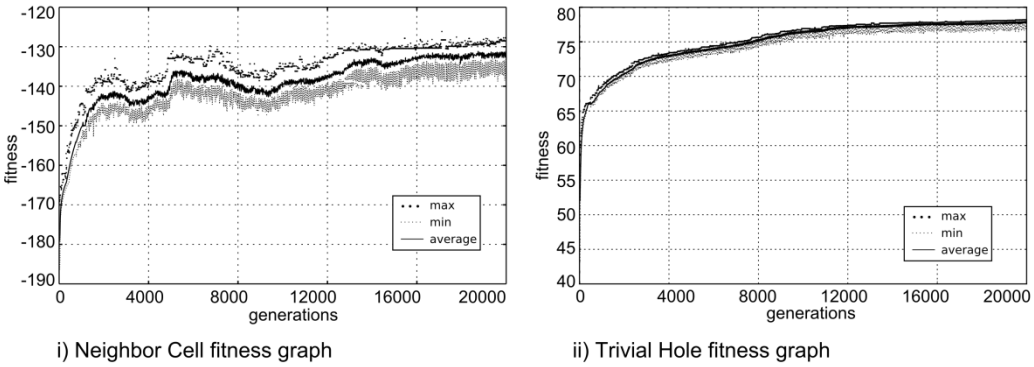


Figure 4.66 : Neighbor Cell + Trivial Hole, example fitness graphs.

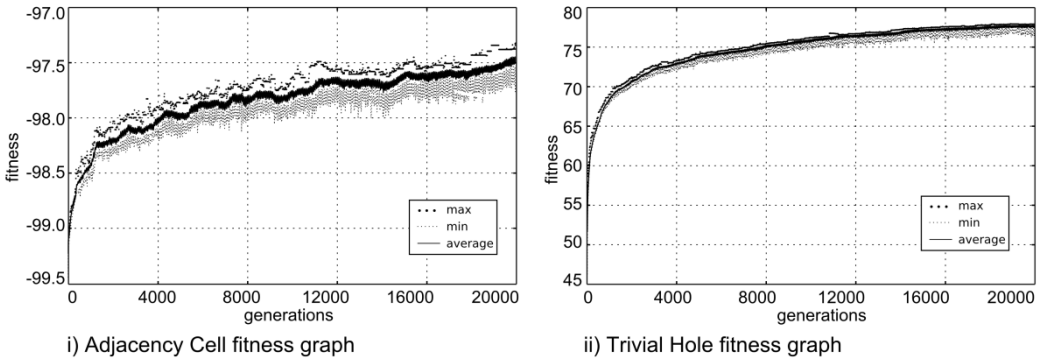


Figure 4.67 : Adjacency Cell + Trivial Hole, example fitness graphs.

Finally, the test setting and example results for the Manual Cell objective can be seen in Figure 4.68 and 4.69. The test tasks are rather easy and as can be seen in Figure 4.70, in an incremental progression (which indicates the attaining of desired proximities one by one), the process converges in just a few hundred generations. Although this test setting is simple (indeed the initiation example in Figure 4.71 has already found the desired cells right through random initiation, which is not usually the case), even the hard tasks for the Manual Cell objective would remain relatively simple compared to the other objectives, and the desired fitness states are generally reached very quickly.

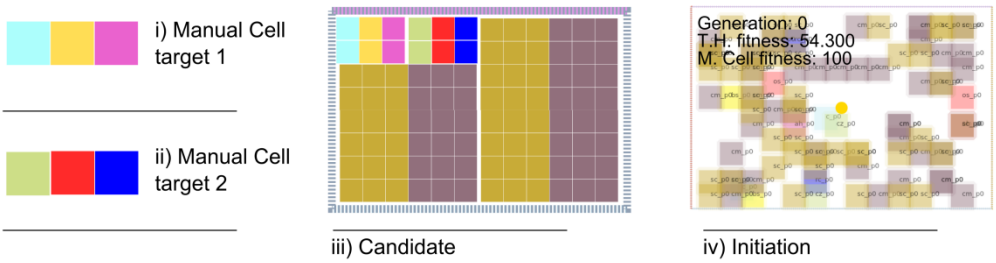


Figure 4.68 : Manual Cell objective; target, candidate, and initiation.

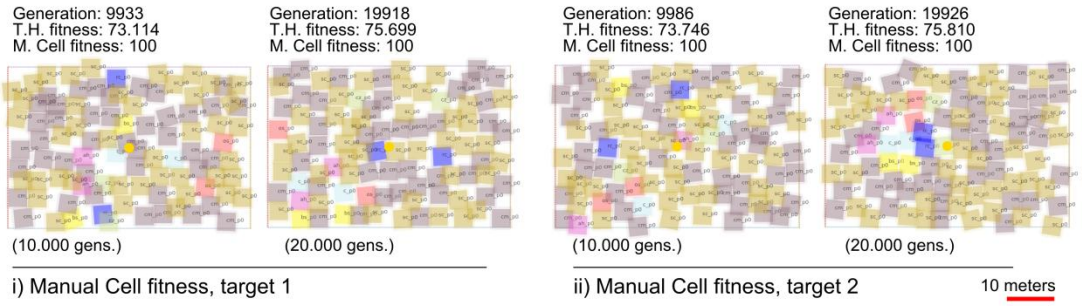


Figure 4.69 : Manual Cell objective, example best results – Grid width: 4m, query cell width: 8m.

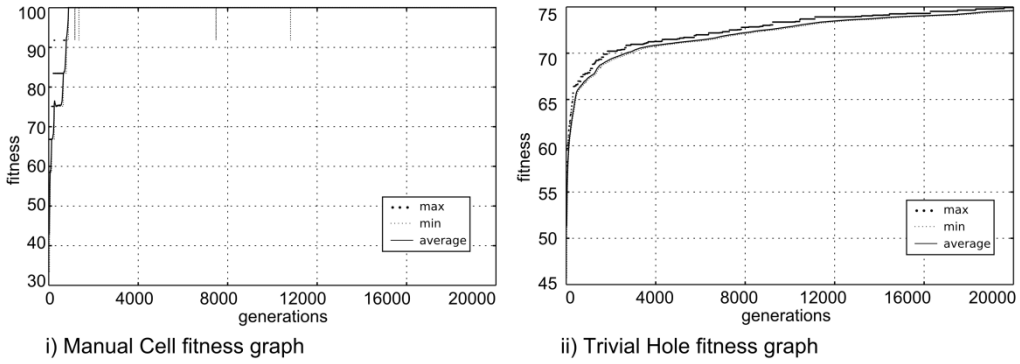


Figure 4.70 : Manual Cell + Trivial Hole, example fitness graphs.

There are a number of critical parameters for each objective, e.g., grid width for the Sequence objective, query cell width for cell type objectives, and DU offset for the Proximity, Neighbor, and Adjacency objectives. The default values for these parameters have been found through trial runs; seeking a balance between computational performance and quality of results. However, best combinations for these may be changing through different scenarios, which is not easy to analytically determine. Specific experimentations have to be devised in order to examine the possibilities with regard to potential scenarios, which would become a separate study on its own. Such an attempt would be a premature fine-tuning for the rather elementary state of the d_p.layout system. Therefore, systematic tests for these parameters have been omitted.

4.2.8 Multi-objective test series

Before systematic tests, the effect of the number of objectives was examined through prior runs for all combinations of 50, 150, and 250 DUs; 3, 6, and 9 objectives; and fully-adaptive, limited-adaptive, and non-adaptive cases, which gives a matrix of 3x3x3 = 27 tests. Due to performance limitations, only a few tests could be carried

out for each case. It should be noted that these limitations were due to unstable hardware and limited RAM, rather than prohibitively long evolutionary runs, which was not the case. Although not systematic, therefore not extensive or certain, these tests indicated that, an increase in the number of objectives, without a parallel increase in population and variation sizes, results in a tendency to converge earlier, compared to trials with a lower number of objectives. This tendency, which is by no means unexpected, is more evident between 3 and 6 objectives, and less between 6 and 9 objectives, which is also due to the fact that the added objectives pursue similar goals with the existing ones in the case of an increase from 6 to 9 objectives. The important point here is not the lower levels of attained fitness but the earliness of convergence. This means, each new objective increases the complexity of the problem, and simply lengthening the process to allocate more time to each objective will have no positive effect. Before concluding on this issue, it should be noted that systematic tests have to be carried out to see the effect of increased population and variation number combinations, which might solve the problem. For the aims of this thesis, we can simply move around this issue and just limit the number of objectives to 3 or 6, which is sufficient for our aims. Time and effort has been saved for other aspects of the problem, while deciding on the issue of higher number of objectives is left for subsequent studies.

The aspects related to the Interleaved EA and the combinatorial operation of objectives have been assessed with the multi-objective trials. Seven different system scenarios have been prepared for comparison. Each of these scenarios has been run 100 times for 5000 generations. All trials involve the six core objectives (Trivial Hole, Out + Overlap, Neighbor, Sequence, Neighbor Cell, and Manual Cell) and 50 DUs. In order to be able to carry out this large amount of tests, the population number has been reduced to 50 with a mutation number of 5. These numbers are indeed quite low, given the complexity of the problem. In Figure 4.71, the average fitness graphs of these scenarios are given as superposed per objective. Each color plots a different scenario as indicated in the legend below the figure. Each line presents the average progression of mean fitness, in other words it plots the average of mean fitness graphs of 100 runs of a scenario. From the progression of the mean fitnesses (Figure 4.71), it can be firmly claimed that all six objectives reliably work together in the multi-objective case.

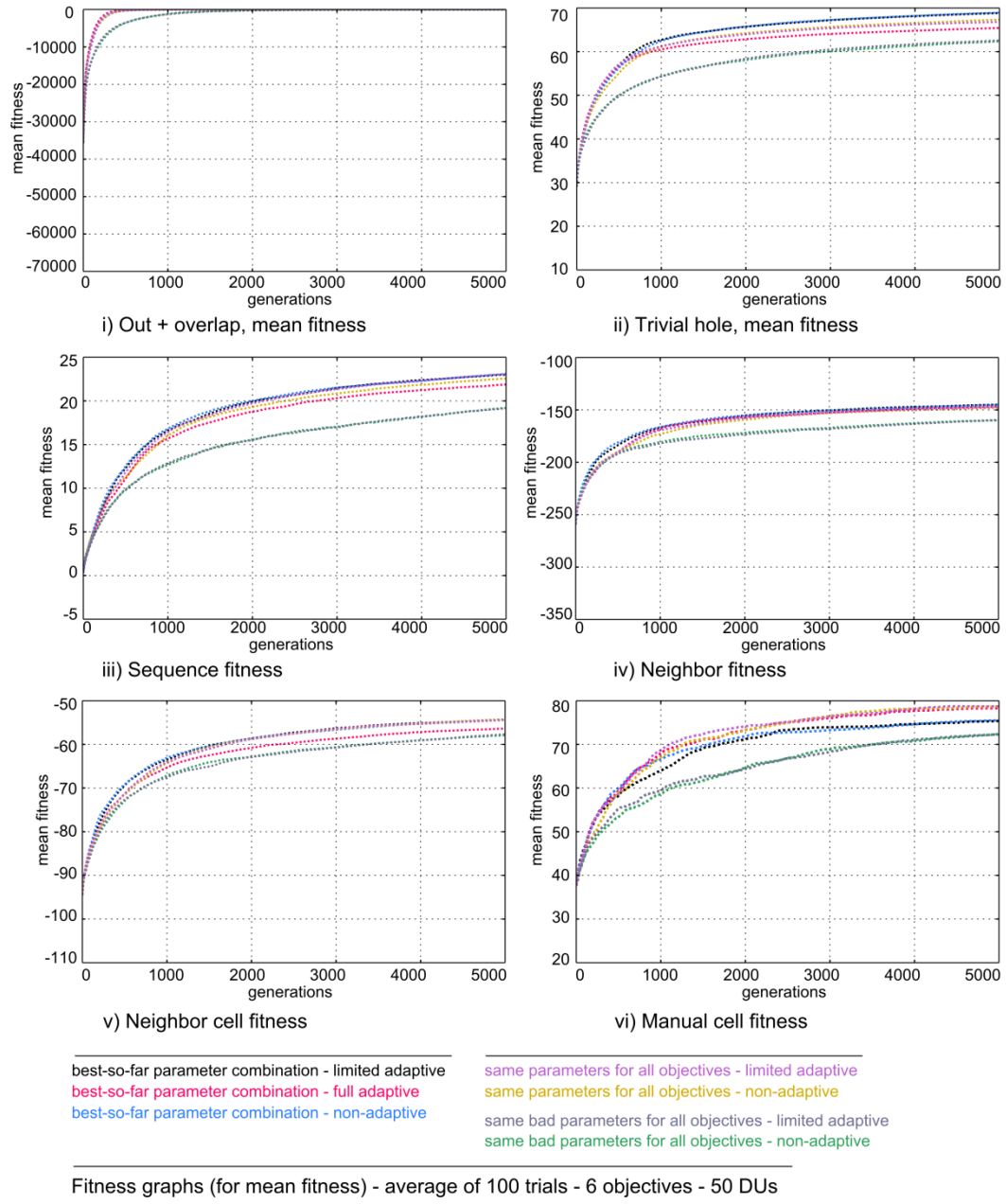


Figure 4.71 : Average mean fitness graphs for the multi-objective case – 7 scenarios, 100 runs for each – 6 objectives, 50 DUs, population: 50, mutation: 5, crossover: 0.

The first issue to be considered will be the effect of adaptive parameter control, which is important for alleviating the need for parameter tuning before each new application scenario. The best test for adaptivity appears to be a comparison of two sets of trials with adaptive and non-adaptive settings, with a parameter combination whose performance is known to be poor. The adaptive version is supposed to perform better, through finding the appropriate parameters by itself through the process. In Figure 4.71, the limited and non-adaptive trials with "same bad parameters" illustrate this issue. In these cases, all objectives are started with the

same poor parameter combinations for process, variations, and adaptivity. It can be seen that there is almost no difference in the fitness progression course of these cases, which shows that for the d_p.layout application adaptivity did not have any advantage.

Another set of tests have been carried out with the "best-so-far" parameters. This means that all objectives have their own fine-tuned parameter combinations for process, variations, and adaptivity. In these tests, three versions of adaptivity have been tested to see if adaptivity is disadvantageous. It can be seen in Figure 4.71 that the fully adaptive case, in which almost all parameters have been set to adaptively evolve, is worse than both limited-adaptive and non-adaptive cases. The limited-adaptive settings have been enabled by the Interleaved EA approach. For each objective, separate adaptivity settings have been developed through both trial runs and a priori considerations of the specific properties of that objective function. Results of the limited adaptive case almost totally coincide with the non-adaptive case, which again gives us no evidence for claiming an advantage of adaptivity.

It should be remembered that advantages of adaptivity had been demonstrated in the previous d_p.graphics application. The most important difference between the two applications is the number of parameters, i.e., the relative complexity of the d_p.layout application. The effect of an increase in the number of evolved parameters can be seen in the decreased success of the fully adaptive case, which shows that a proliferation of evolved parameters makes the problem more complicated and this decreases overall performance. These systematic test series have been carried out with low population (50) and mutation (5) numbers, and for a short generation count (5000). These settings may be too low for a complex adaptive case to deploy its advantageous properties and some properties may be activated only in the long run. Nevertheless, even with these settings, a superiority on behalf of the adaptive version was expected; although a minor one. It did not appear.

As a second potential explanation, the failure of adaptivity may have resulted from the adopted adaptivity approach. In the d_p.layout version, whenever an individual is selected for mutation, first, some of its adaptive and self-adaptive parameters are mutated. And at the same turn, right after the mutation of parameters, the individual is mutated according to the parameters that are specific to that individual. In other words, at the same run, first parameters are mutated, then with those new parameters,

the individual. An alternative approach could be to select two distinct pools of mutation candidates. While one pool goes under process parameter mutation (i.e., adaptation) the other would be mutated in the regular sense. Thus, there might be problems in the adaptivity mechanisms or in the parameter settings of the trial runs. These have to be assessed in further studies. Within the span of this thesis, the limited-adaptive version is kept as the default option, for it appears from the tests that it is not disadvantageous, either.

The second issue to be considered through the systematic tests is to demonstrate whether the rank-based Interleaved EA approach is better than a regular rank-based scheme. To test this hypothesis, a setting is devised, where adaptivity is totally disabled and all objectives are started with the same parameter combination. It is difficult to claim that these parameters are the best-so-far parameters, for best parameters vary for each objective (which can be claimed to show a priori that the Interleaved EA is preferable). Nevertheless, a balanced parameter combination is devised as much as possible. In addition, a limited-adaptive version of this same parameter combination is tested. As can be seen in Figure 4.71, it is not possible to claim a clear superiority of the two same-parameters versions over each other. However, with the exception of the Manual Cell objective, which is distinctive in general, the same-parameters cases are either worse or the same with the non-adaptive and limited-adaptive Interleaved EA cases. Thus, it can be claimed that the Interleaved EA, with its objective-wise parameter specification, is advantageous over both of the same-parameters versions. It should be noted that, all tests have been carried out on the same computer and we could not detect any consistent difference in the time expenditure of different trial cases. Therefore, this superiority comes with no performance cost. As a result, it can be claimed that the Interleaved EA is preferable.

In summary, the functionings of all objectives have been shown to be working mostly as expected, although in experimental conditions and not in multi-objective settings. The reliability of the multi-objective operation of six of the objectives is also demonstrated with the systematic trials. It appears difficult to visually demonstrate multi-objective functioning because of the difficulties in distinguishing the effect of each objective within a combined case. Nevertheless, this will be attempted with regard to example applications in the following pages.

The tests were successful in showing that the Interleaved EA has advantages over same-parameter approaches. However, we were unable to demonstrate an advantage on behalf of adaptive parameter control.

Although it would be interesting to experiment with Pareto-based alternatives, a comparison with Pareto-based approaches is missing in this application, which is also left for future studies.

Before finishing this section, an example multi-floor layout evolution will be illustrated. The initiation layouts of the five-floor building are given in Figure 4.72. For this example problem, in addition to a few fixed DUs, a list of rectangular DUs is given to the system, which distributed these to five floors using the information from all six available target buildings (which are given in the Appendices). Each floor receives around 30-35 DUs, which appears as a reasonable amount for a medium sized building. (Automated DU list generation functionality will be exemplified with the applications later in this chapter.)

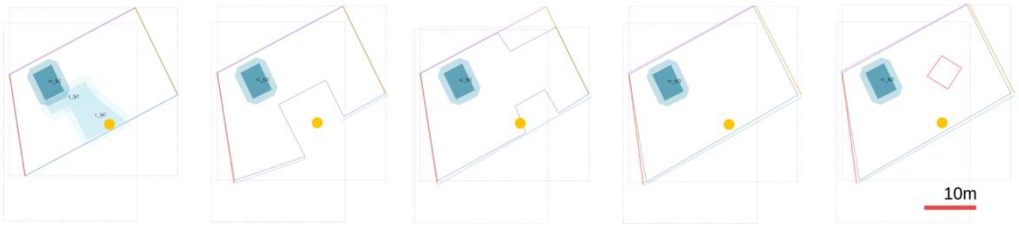


Figure 4.72 : Prepared outlines and fixed DUs of a five-floor example run.

A vertical circulation area is fixed on all floors (Figure 4.72). In addition, an entrance hall is fixed on the ground floor. The blue outline gives floor boundaries and colored lines define directions. The 'indent' within the boundary of the first floor is thought as the continuation of a two-storey high entrance area. Boundaries of the remaining floors are also diversified to see the effect of differing conditions. The red square on the fourth floor is a terrace courtyard.

The target library for the Neighbor Cell and Neighbor objectives is Santa Monica Library (can be found in the Appendices), because it is a large library that includes ample information regarding these aspects. The library addition of the Free University of Berlin is assigned as the Sequence target. This library has an exceptionally monotonous spatial organization, where bookshelves, circulation elements, and open study areas are placed in specific sequences, which has the

advantage of legible organization, and is generally reflected in the results (Figure 4.73). However, this library includes almost no sequence information for most DU types, which is a serious drawback against real world applications. Finally, Manual Cell target is given as "[vertical circulation, technical space x3, WC x2]".

Each floor has been assigned a maximum of 10000 generations, which has only been completed by floor 4. A population size of 150 is used, while mutation and crossover numbers were set to 15 and 0 respectively. Crossover was by default disabled because of the nature of the problem, in which absolute placement of a DU did not matter as much as relative positioning; crossover would amount to teleporting a set of DUs to random locations. Fitness graphs of the ground floor (floor 0) can be seen in Figure 4.73.

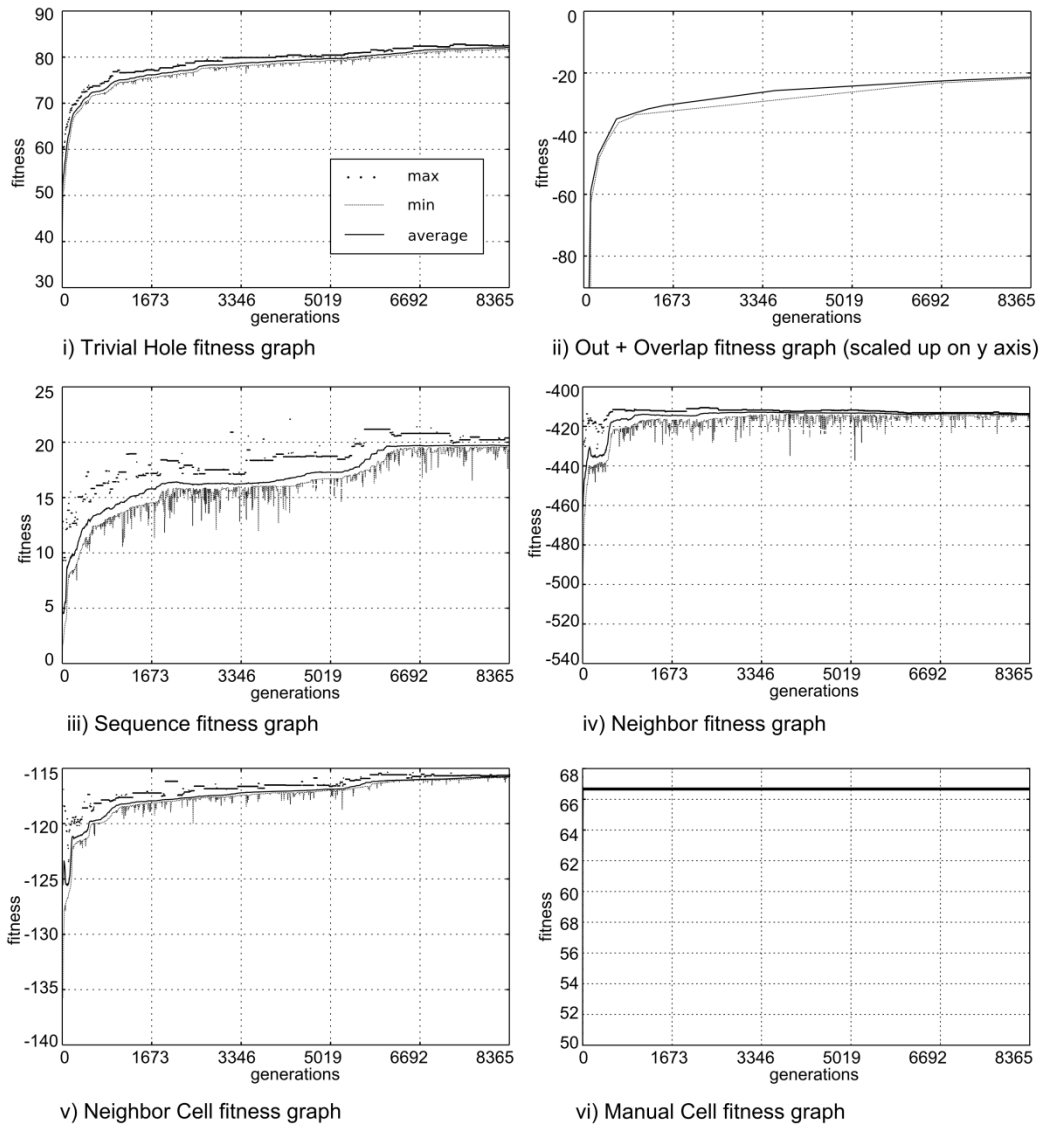


Figure 4.73 : Fitness graphs for the ground floor of the example run.

For each floor, the system plots either the best individual for each fitness, or highest rank individuals, or both. The user may inspect the last few plotted examples to determine preferred results. However, a convenient way is simply to take the last plotted example, because the system tends to raise the average of all fitnesses, so that it converges around similar examples and different best-result options frequently give the same results. The chosen results of this example run are given in Figure 4.74.

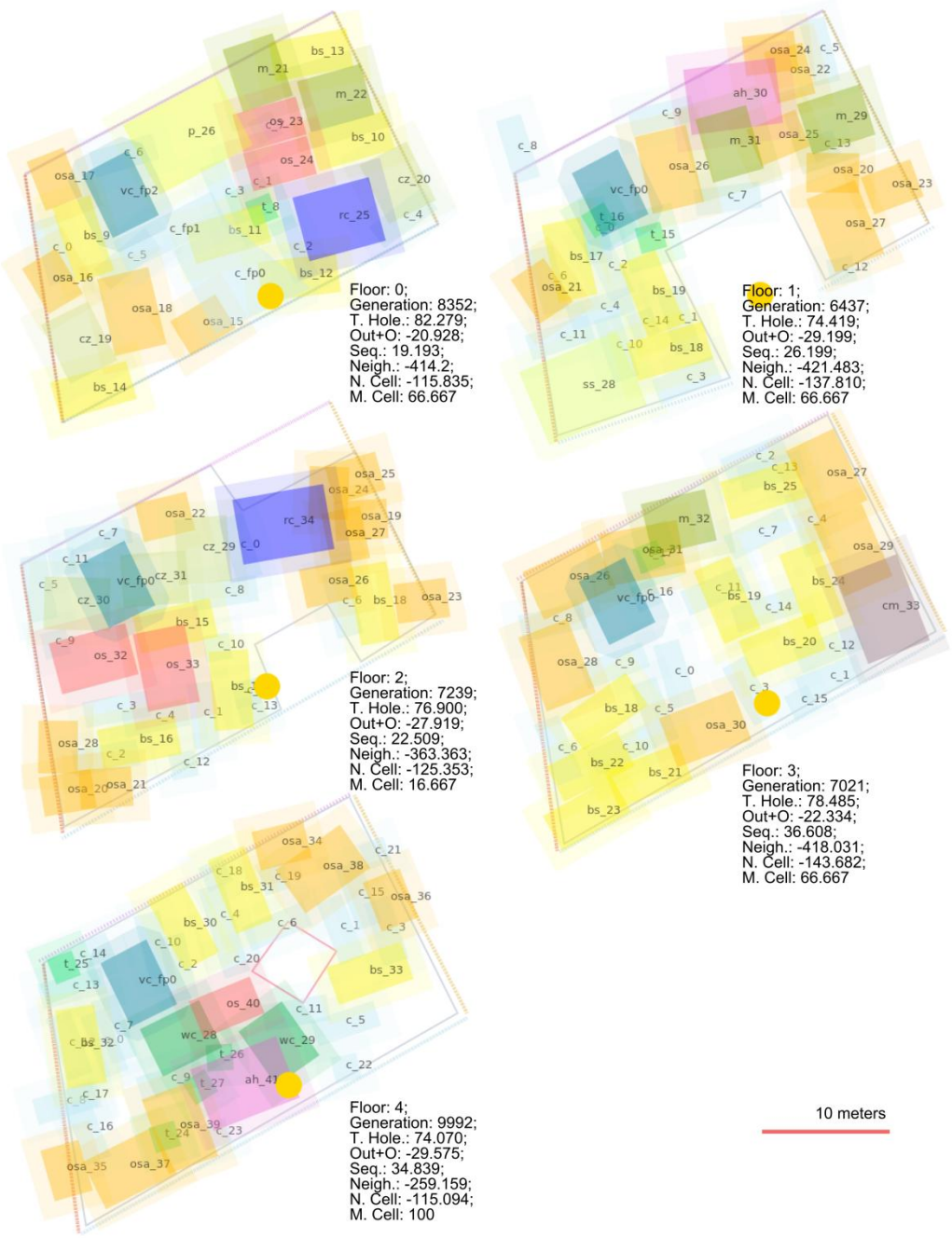


Figure 4.74 : Resulting best floor layouts for the example run (max. gen.: 10000, pop.: 150, cross.: 0, mut.: 15, Manual Cell target (("vc", "t", "t", "t", "wc", "wc")), Neighbor Target = Neighbor Cell target: Santa Monica, Sequence target: Free University Berlin, using max. %92 of the available plan areas, DU offset: 1.2m, Sequence grid width: 1.5m, M. Cell grid width: 8m, query cell widths: 16m).

From the results in Figure 4.74, it can be claimed that the DU types are distributed to the floors in mostly an acceptable manner. Office spaces (os – red) are distributed to several floors in mostly central locations. Library functions, i.e., bookshelves (bs – yellow), open study areas (osa – orange), multimedia areas (m – dark green), and

comfort zones (cz – light green) are distributed evenly to the floors. Because of the Neighbor and Neighbor Cell target, i.e., the Santa Monica Library, certain DU types like open study areas and bookshelves tend to get close together. This target is highly organized and it strongly separates functions into zones (compare with Halmstad Library's complex organization in Appendices), which is reflected in the results. In addition, open study areas can be seen to prefer floor boundaries, which is mostly due to the Sequence target, which located these to floor edges almost without exception.

Likewise, targets caused the bookshelves to concentrate mostly near each other, but these are also evenly interleaved with circulation elements. This is due to the regular organization of the sequence target. Because of the Manual Cell objective, core functionalities, i.e., technical spaces (t – bright green), WCs (wc – dark green), and vertical circulation (vc – dark blue) tend to come together. Activity areas (ah – pink), restaurant-cafe's (rc - blue), commercial area (cm - purple), and periodicals (p – lemon yellow) had to be arranged with less information, which clearly shows a requirement for methods that would enable the collective usage of a series of targets together for each objective, as is the case with DU distribution to floors.

Formal objectives were more successful on simpler floor shapes. DUs have to adapt both to each other's borders and to floor borders in a collective manner. Initial DU rotation and subsequent arranging is much easier for simpler boundaries. Nevertheless, even the intricate floors with several large DUs have evolved to an acceptable degree.

It can be claimed with this example that, although its capacities are seriously limited, d_p.layout is able to evolve draft layout arrangements. It should be thought of as a starting point with a minimal set of capabilities. It appears that, for more meaningful applications, usage of basic zoning information is also required, such as with regard to orientation, access, views, and functionalities. This information is indeed present (i.e., painted) in the target examples; however, specific techniques (i.e., objective procedures) have to be developed to utilize this information, which is left for future studies. Additionally, for more complicated problems with greater DU numbers, a hierarchical approach appears necessary for dividing and guiding the process.

4.2.9 Validation of the design_proxy.layout

For an assessment of the capability of the design_proxy.layout system to be incorporated within design processes, in the following sections, two workshop processes will be described in detail. After a general introduction into the aims and procedures, specifics will be detailed for the two workshops. For each of the workshops, two example design processes will be illustrated to discuss the capabilities and limitations of the d_p.layout in practice. After these detailed descriptions, the last section will involve an overall evaluation of the d_p.layout. This evaluation will start with an exposition of the evaluation criteria and will continue with an assessment of whether these criteria have been met or not. This assessment will be extended with an evaluation of the participant surveys. The chapter will be closed with a general consideration of the basic utility of d_p.layout in design practice.

The most important amongst all considerations towards the d_p.layout system is its potential for being used within real world situations by human designers. This is required not only to show that it is readily useful, but also because this is the only way forward for further development, from the system's current experimental situation to a more developed professional-quality application. To begin with, human usage is required for preparing and bringing together target libraries for a range of example architectural typologies, hence for collecting the required raw information. In this context, a community based open project would reveal the highest potentials for further development. Yet, the more important point is that the system has to be developed through considerations of user feedback, and eventually through real-time feedback with the help of machine-learning technologies.

These aspirations can be realized only through the system's viability at the outset. This means, such a system has to lift itself from its bootstraps through some sort of congeniality and usefulness and situate itself on a path of symbiotic relationship with human users. In brief, d_p.layout has to be demonstrated as technically functioning, practically useful, easy to use, and pleasurable enough for use. With regard to these aspects, we have to find out if the system has a viability to be incorporated in design processes.

The technical competence of d_p.layout has been examined in the previous sections, where it was shown to carry out its tasks as expected while its capabilities were primitive. However, the initial requirement for an artificial design assistant is not technical competence, which is not possible with the available AI technologies. Currently, no artificial agent is competent enough to be compared with humans for carrying out architectural design. Yet, the only way to move forward to that direction is to situate AI systems within design practice, which requires us to extract a kind of usefulness even from such primitive systems, or alternatively, to devise systems that would be useful even within such technical limitations.

Several other questions appear at this point regarding d_p.layout: "Is the system versatile enough to be readily applied to a wide range of scenarios?" "Is it flexible enough to let its user devise ways to exploit it?" "Is the system's interface easy enough to learn, adapt to, and utilize?" "Is it easy to interact with?" "Are the outputs of the system easy to interpret and put to use?" And "what kind of benefits might be expected from such a system in terms of speed, quality, and pleasure?"

These questions have no ideal answers and for an attempt at answering them, the system has to be put to use within real design processes that have to be experienced and interpreted. For this reason, in this section, two student workshops will be illustrated to discuss whether d_p.layout is sufficient for some usage scenarios according to this variety of aspects. The workshops have been carried out at two universities, which differ in educational approaches, academic staff, and background and ages of the student groups. This raises both a difficulty for comparison and a potential richness of interpretation.

Both workshops have been shaped and presented as regular architectural design workshops, i.e., not as Computational Design workshops. Although d_p.layout was being introduced as a participant of the process, the focus was on the design of library buildings, at specific sites, by the students, and in a collective manner. It should be noted that the specific structure of the workshop processes were as effective as the d_p.layout in the formation of the results. After all, these workshops tried to situate the d_p.layout within design processes, not the other way around. The core design steps of the workshops comprised fast collective brainstorming sessions, study, presentation, and discussion of precedents, exercises that spatially enrich resulting buildings, and a fast progression towards a detailing of 1/200 models. As

will be illustrated in the following sub-sections, the d_p.layout has demonstrated its utility in all these core steps.

It should be noted that the students have not specifically chosen to attend the workshops, and they have not been chosen amongst a range of applicants. The workshops were embedded within regular courses and they have been carried out alongside regular design studios. This has made attendance more difficult, yet the degree of attendance was satisfactory.

For the workshops, the d_p.layout system has been carried with all its dependencies in a ready-to-launch state on live USB devices that run the Ubuntu operating system (licenses of all dependencies permit non-commercial redistribution). This enabled the rapid conversion of regular computers in the computer labs to d_p.layout stations to be controlled through the internet with parameter files stored on shared folders. Again, shared folders were used for accessing the outputs of the trials, which has the benefit of easy access by the students, together with automatic archiving.

The basic document package included the workshop posters with a workshop statement, site maps, graphical presentations of target buildings, DU type legends in the form of bookmarks, student surveys, color sheet packages, and d_p.layout parameter sheets (which were never used). Additionally a digital package of these documents and Inkscape template files were also given to the students.

4.2.10 Evolutionary Collectivity İzmir workshop

The first workshop has been carried out from 4th to 8th of March 2013 at Yaşar University in İzmir, Turkey. It was prepared and tutored by the author together with Ahu Sökmenoğlu (İTÜ, TU Delft) and Ioannis Chatzikonstantinou (YÜ, TU Delft). The progression of the five-day workshop process is depicted in Figure 4.75.

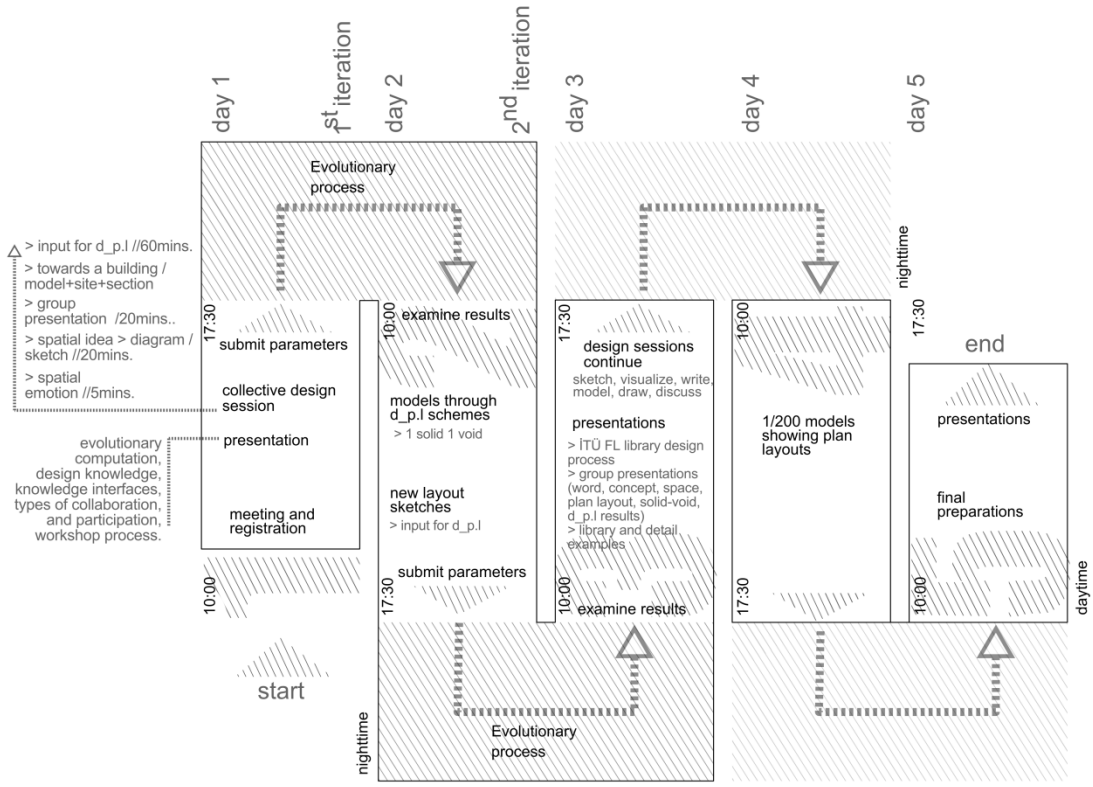


Figure 4.75 : Workshop process for Evolutionary Collectivity İzmir workshop.

With a few exceptions, most of the participants were 4th year students. In total seven teams (14 students) finished the workshop. Additionally, four graduate students visited the workshop for one day to experience a very fast version of the workshop workflow, but they did not continue to completion.

In the opening presentation, operation principles, capabilities, and limitations of the d_p.layout system were introduced along with background information on evolutionary computation. Limitations of the system were especially emphasized to prevent over-reliance on the system's output, in an attempt to establish a balanced and sound human-machine interaction.

The workshop had been planned with two iterations of interaction with the d_p.layout. Initially, the first iteration was thought of as an introduction to both the problem and the d_p.layout system, while the second iteration was expected to be more productive. It turned out that this was up to how each team was approaching its problem.

The design process started with a collective brainstorming session (see Figure 4.76 for workshop photographs). Through several stages, from concepts expressed with a few words, the students moved forward to sketched out and diagrammed spatial

ideas. The process started with the participation of the whole group. Further on, the participants were divided into groups of eight, and eventually they continued the workshop individually or as teams of two to three people. At this stage, they were asked to choose their favorite spatial ideas and start working on the main problem of the workshop; that is, designing a medium-scale library within the given site. The site for İzmir workshop was right across Yaşar University Campus, and was easily accessible. It was almost flat, in between a metro line and a highway, next to a metro station and was being crossed by a pedestrian bridge, so that there were many situational parameters in need of reconciliation.

Figure 4.76 : Images from the Evolutionary Collectivity İzmir workshop, from brainstorming sessions to final presentations.

The ultimate task for the first day of the workshop was to determine a massing within the given site, with regard to the contextual aspects, the given problem, and the conceptual and spatial ideas that were being gathered through the initial exercises. The convenience of working with d_p.layout first appeared at this point. The

decisions of the students have not been criticized, so that the brainstorming process would continue to include the d_p.layout's contribution. Therefore, even the least interested team was able to put a simple mass somewhere inside the site, just to see what comes out from the system. The teams then horizontally sliced their masses to obtain initial floor outlines to feed to the system. At this point, the tutors imported these sketch plan outlines into Inkscape, painted these as candidate layouts, parsed them and started d_p.layout to run through the night for each team. Originally, this painting operation was to be carried out by the teams; however, because the Inkscape interface was not fail-safe, first testing of the system has not been put at risk. In the second workshop in Mardin, the interface has been used by the students, where a short Inkscape course sufficed to prepare the students for the task.

The next morning, the students came back with the printed results of the d_p.layout, which they have found in the shared folders. Some trials with smaller layouts had accomplished more than one run, while a few had not been able to finish all floors due to large layouts and high amount of DUs. Although technical problems occurred in a few examples, all teams acquired a usable response from d_p.layout, either well developed or not. The resulting d_p.layout graphics were discussed together to develop an understanding on what they mean, and how they may be interpreted and used.

Now that they had draft DU arrangements filled within their given outlines, the students became ready to get involved with the spatial and functional organization of their proposals. At this point, in order to study the vertical and horizontal circulations and to elaborate the visual and spatial connectedness of the interiors, an exercise was given, according to which two physical models had to be prepared, one solidifying the voids, i.e., visualizing only circulation spaces and galleries, and the other for solidifying the used spaces on the d_p.layout outputs. Although a bit misleading, these models were shortly called "solid" and "void" models. Obviously, this exercise is not a necessary constituent of the d_p.layout workflow; yet, it is fully congruent with it, because the d_p.layout outputs include large interconnected circulatory areas and both intentional and unintentional empty fields. This exercise gave the students an incentive to study d_p.layout's proposals and to connect their spatial studies with it. It also greatly contributed to the interior spatial quality of their final proposals.

Through the study of the d_p.layout drafts, the teams entered into functional considerations much earlier than they would, at least with such detail and comprehensiveness. Indeed, entering into functionalities right from the beginning could be considered premature. However, it was not exactly the students who delved into functional studies, but the complex, constituted by the teams together with the d_p.layout, where the burden of the details was initially only carried by the machine counterpart, which leaves the human mind free for conceptual considerations.

It is true that the d_p.layout yields DU arrangements, but first of all it develops building programs in the form of DU lists. Although these lists are by no means refined programming studies, they are at least probabilistically drawn from collections of existing buildings. Thus examining a few of these layout proposals quickly gives the student a feeling for the basic functional requirements of a specific typology in a rather painless way. It should be noted that, because the study is carried out over the functional arrangements of the d_p.layout, the solid – void exercise fluently connects spatial, conceptual, and functional studies, and quickly brings the students to a stage where they can start better investigating functional arrangements together with spatial effects. In this sense, this exercise greatly increases the speed of the workshop; however, this acceleration cannot be solely attributed to the d_p.layout. In brief, if speed amounts to the omitting of contextual and conceptual issues, it may result in premature products. However, in this case, it rather meant more time and a more integrated process for the contextual, conceptual, and spatial aspects.

The following step in the second day of the workshop was to sketch over the initial results of the d_p.layout to refine the layout arrangements along with new model and section studies. Through this step, some of the teams prepared a second proposal to be given to the d_p.layout. However, several teams had already entered into a productive path or structured and confined their proposals to such a degree that the d_p.layout had nothing left to offer. It should be remembered that the d_p.layout is not a layout refinement tool; it is only for initial patching studies and is most useful in the beginning of a design process. A second iteration means repeating the same step but with a better understanding of the problem and the system, and through more refined and controlled proposals.

The third morning started with the examination of the second set of results. The students brought with them architectural examples and presented their two-day progressions together with these examples. Such examples have a very important role in the progression of a design workshop, and both the tutors and the student teams have brought relevant examples, which were examined and discussed collectively. Additionally, Ahu Sökmenoğlu presented a small library project, which had recently been implemented by her. This presentation was also important and provided the students with a close-up example of the rationales and processes of library design and application. After these presentations, the teams started to work with regular design tools such as sketches, drawings, and models, and the tutors provided support and desk critics. The ultimate output would be 1/200 models with interior organization depicted by the same color-coding scheme that is used for the d_p.layout and the targets.

Overall, the workshop was perceived as satisfactory by both tutors and observers (impressions of the participants will be discussed in the following sections over student surveys). As expected, the finished projects varied in quality, yet they were all sufficiently detailed, displaying rich interior and exterior spaces; more developed than would be expected for such a short period. The d_p.layout was so integrated in the process that it is not always easy to distinguish its specific effects, yet it is easy to trace its integration.

Again as would be expected, there were both less interested students and highly involved ones who kept up better with both the workshop process and the d_p.layout. More involved teams tended to accept the d_p.layout as a constituent of their design processes. They first cooperated and then started to play with it and managed to make it a part of their conceptual and spatial studies. However, this progression was not observed with all teams. Apparently, some students tended to perceive the d_p.layout to be contesting their habitual way of design, and did not prefer to fully adapt to the workshop process. The outputs of the second type of students tended to be lower in quality. This was caused by a combination of a lack of attention and a dissonance with the workshop's progression, which tended to reward cooperation with the d_p.layout. The trust of the students towards the tutors and the system has an important effect in the outcome. However, this trust should be kept in balance, because the student should not over-emphasize the capabilities of the system and

should remain the real author of the process. In the beginning, the students tend to approach such a system as if behind a mystical veil, but with the inspection of the first results, they start to better understand its actual capabilities. If the students start to think that the d_p.layout is capable of bringing out usable drafts, and at the same time start to trust the tutors, the constituents of the process start resonating and the students get ready to accept the d_p.layout and start playing with it. It should be noted that interest, sincerity, and a liking for learning and producing are more effective than any tool. It appears that these are the assets that make tools work, not the other way around. The workshop process was designed to facilitate a convergence between the d_p.layout and its users. In other words, the design process was tailored for the d_p.layout, in a way that it would appear less demanding and more congenial to humans so that they would approach it. Therefore, neither success nor failure of this process can be totally attributed to the d_p.layout. Yet, the d_p.layout appears capable of joining in this process.

It is difficult to assess the contribution of the d_p.layout for less involved teams. Therefore, the workshop processes will be illustrated over examples, which are not claimed to be typical. Example processes are chosen for their stronger interaction with the d_p.layout. The legend for the simplified color-coding scheme that was used for the two workshops is given in Figure 4.77.

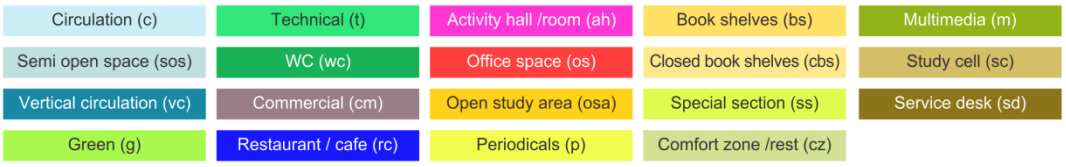


Figure 4.77 : Simplified color-coding legend.

The first example is the team of Müge Halıcı and Ceren Abacıoğlu, who were 4th year students at the time of the workshop. They started their examination with the keywords, "non-accessible", "spaciousness", and "airy", which they developed into a conception that understands interior space of a building in terms of "largeness and extensiveness" (Figure 4.78). They then moved forward to question the relationship between "inside" and "outside" and set forth a hierarchical spatial organization from "protected" to "unprotected". At this point, they envisaged their building as a cube that is pierced by the pedestrian bridge that crosses over the site. During the night, the d_p.layout processed seven floors of this large cube. It turned out that the team

made a mistake about the height of the bridge; however, this was no problem at all. The proposal consisted of just floor outlines, and it was the d_p.layout who worked through the night.

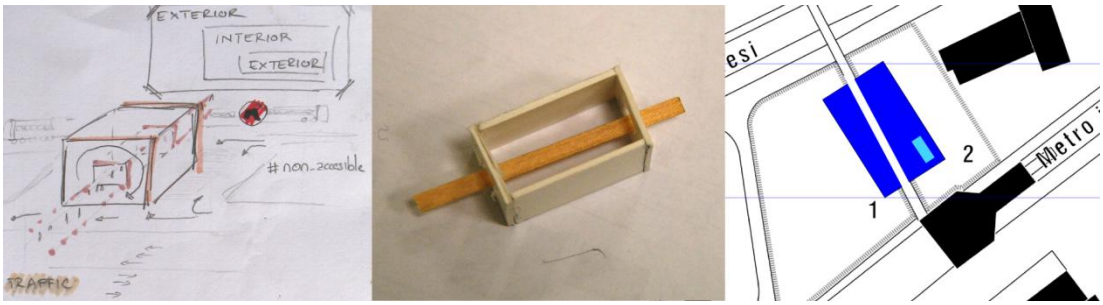


Figure 4.78 : Müge and Ceren – left: initial sketches, middle: first model, right: sketch submitted for d_p.layout.

In the first results for the seven floors (Figure 4.79), the layouts include no vertical circulation elements or distinct circulation areas. Functional zones are not coordinated between floors, yet they involve a sufficient variety of functional elements that are arranged with regard to their functionalities. It is difficult for a human designer to delve into such detail at this stage, because she has to think about a large group of contextual and conceptual factors before moving into detailed arrangements. However, in this case there is almost no cost of bringing out these draft arrangements.

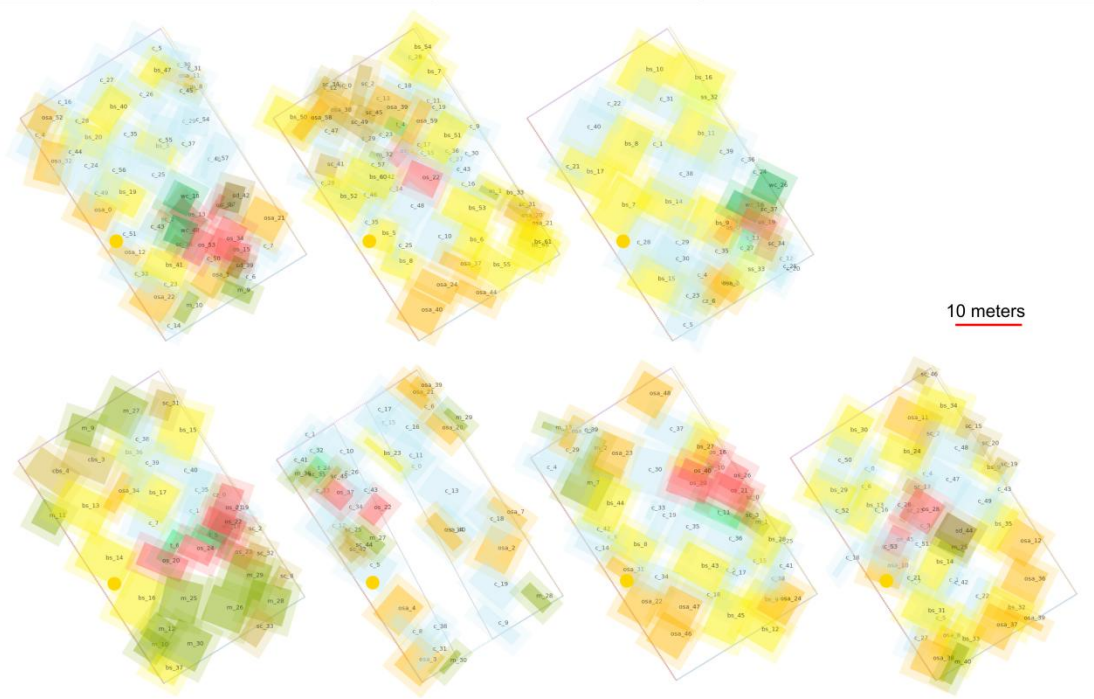


Figure 4.79 : Müge and Ceren, results of the first trial (seven floors, max. gen.: 7000, pop.: 150, cross.: 0, mut.: 15, Manual Cell target (("vc", "t", "t", "t", "wc", "wc")), Neighbor Target = Neighbor Cell target: Santa Monica, Sequence target: Free University Berlin, using max. %92 of the available plan areas, DU offset: 1.2m, Sequence grid width: 1.5m, M. Cell grid width: 8m, query cell widths: 16m, DU area limit: 45m^2, DU width limit: 8m, formal refinement on with mut.:75, diminish: 25)

It can be noticed in Figure 4.79 that the initial DU rotation functionality failed to approximate layout orientations and the DUs are collectively placed in a diagonal orientation. The system fitted lines using the four corner points of the floor outlines, which would be different with another orientation. Whenever a high number of DUs are wrongly orientated, they have to adapt to the floor outlines in a collective manner, because their orientation is also connected to the neighboring DUs. This is a complicated combinatorial task that would most probably be omitted, while the system can maximize its fitness through easier operations towards sub-optimal areas[24].

[24] In later applications, this failure is corrected by using another approach, which uses the most dominant rotation angle of the lines of the initial floor outlines, instead of fitting lines over the corner points.

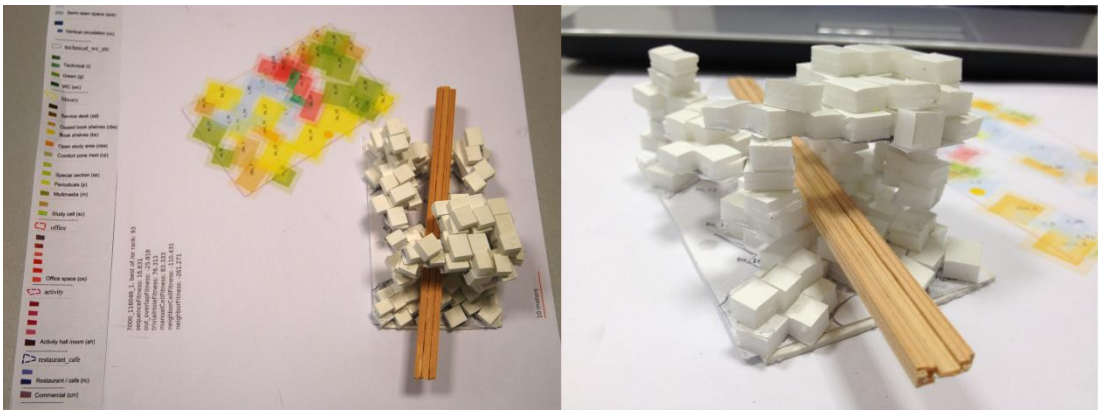


Figure 4.80 : Müge and Ceren, solid-void exercise over d_p.layout drafts.

In any case, Müge and Ceren accepted both the lack of vertical coordination and the failure in orientation as playful ideas for developing their project, which is reflected in their solid-void study (Figure 4.80). At this stage, through a free interpretation of d_p.layout drafts with respect to contextual issues, they prepared a more refined trial, which included locations of vertical circulations, a few fixed DUs, and a variety of intricate insular outlines at each floor level (Figure 4.81). This way they organized their process within a hierarchical progression from rough undetailed outlines to a more intricate state, but still without delving into all the details of the plans. Instead of sequential phases, this progression was realized through holistic 'packages', which differed in their degree of refinement but not exactly in their content. It can be noticed that Müge and Ceren's process never diverted from their initial conceptual investigations on spaciousness, accessibility conditions, and interior – exterior relations. This should be attributed to d_p.layout's non-intrusive manner, which assists by setting forth details and bringing possibilities into mind, while not enforcing any manner of working.

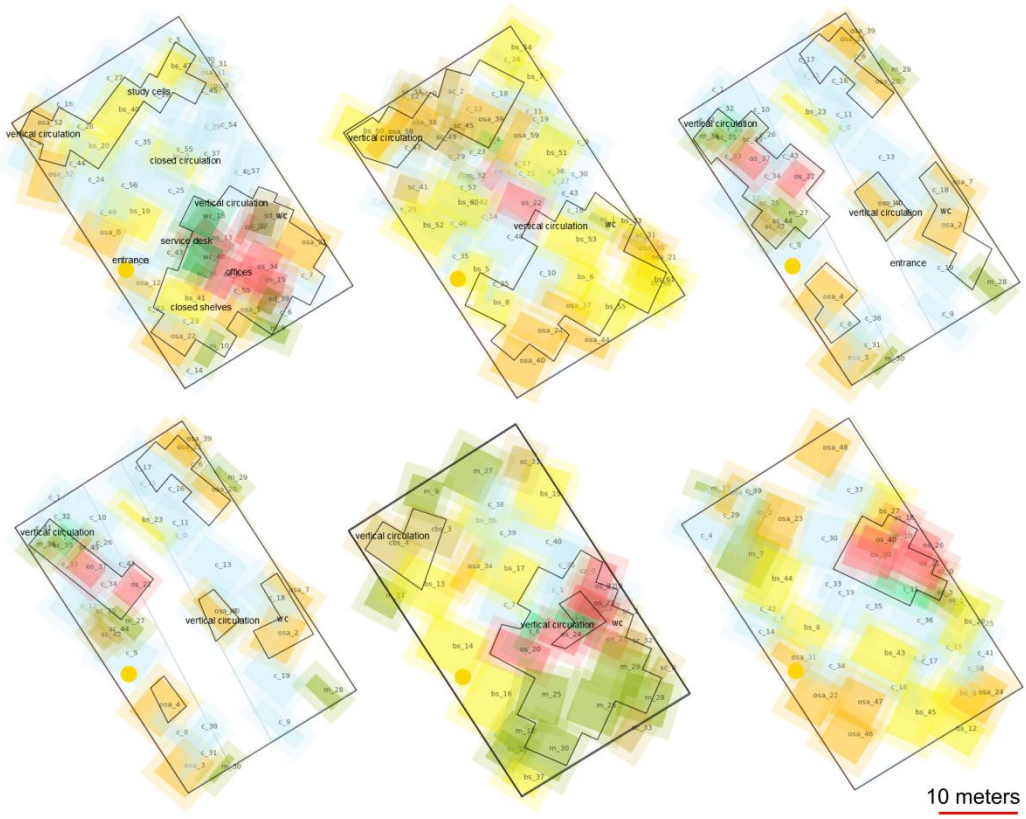


Figure 4.81 : Müge and Ceren, sketches submitted for second d_p.layout iteration.

It can be seen that the two trials are very different in nature. The first trial only included rectangular floor outlines. The second trial (Figure 4.81) involves small islets with complex outlines and continuous vertical circulation, all placed within a simple rectangular building box, which reduces the effect of directions. Results of the second trial can be seen in Figure 4.82. Because the shapes of the layout islets were intricate and narrow, maximum DU width was reduced to 4.5 meters (it should be remembered that actual DU borders should be sought somewhere in between inner and outer borders, which virtually extends this minimum width). The collective DU orientation problem does not appear in the second case because of the high number of vertices that helps in finding the overall orientation of the outlines. This was also because, in the second trial the DU-to-DU relations were less dominant than DU-to-border relations.

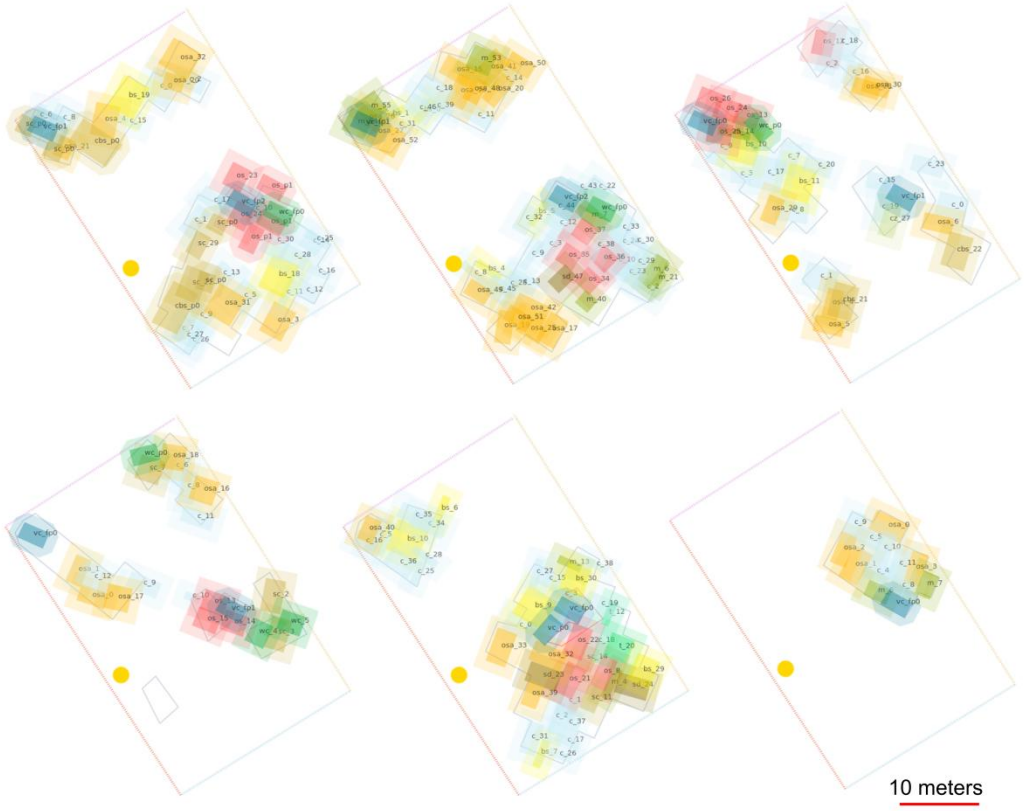


Figure 4.82 : Müge and Ceren, results of the second iteration (six floors, max. gen.: 8000, DU area limit: 30m^2, DU width limit: 4.5m).

With only these two examples, it can be firmly claimed that the approaches underlying d_p.layout result in a highly versatile and robust system that can be applied, with only a few minor parameter adjustments, to cases that are considerably different. This versatility will be observed with all of the following examples.

With the second output from d_p.layout, Müge and Ceren started focusing on their own interpretations. They moved from this stage towards simplifying and rationalizing the multi-dimensional interior organization, which resulted in a well-developed plausible proposal with rich interior spaces (Figure 4.83). The patching colors on the 1/200 model follow the same color-coding scheme given in Figure 4.77. Coloring the model has the benefit of uniting physical models and draft layouts. Although Müge and Ceren have presented technical drawings, the workshop did not require the participants to draw detailed plans, for d_p.layout would not be able to assist users at those stages. The aim was rather to obtain an overall zoning in terms of DU types, which appears sufficient, or even better, for evaluating a specific proposal at this early stage of the design process.

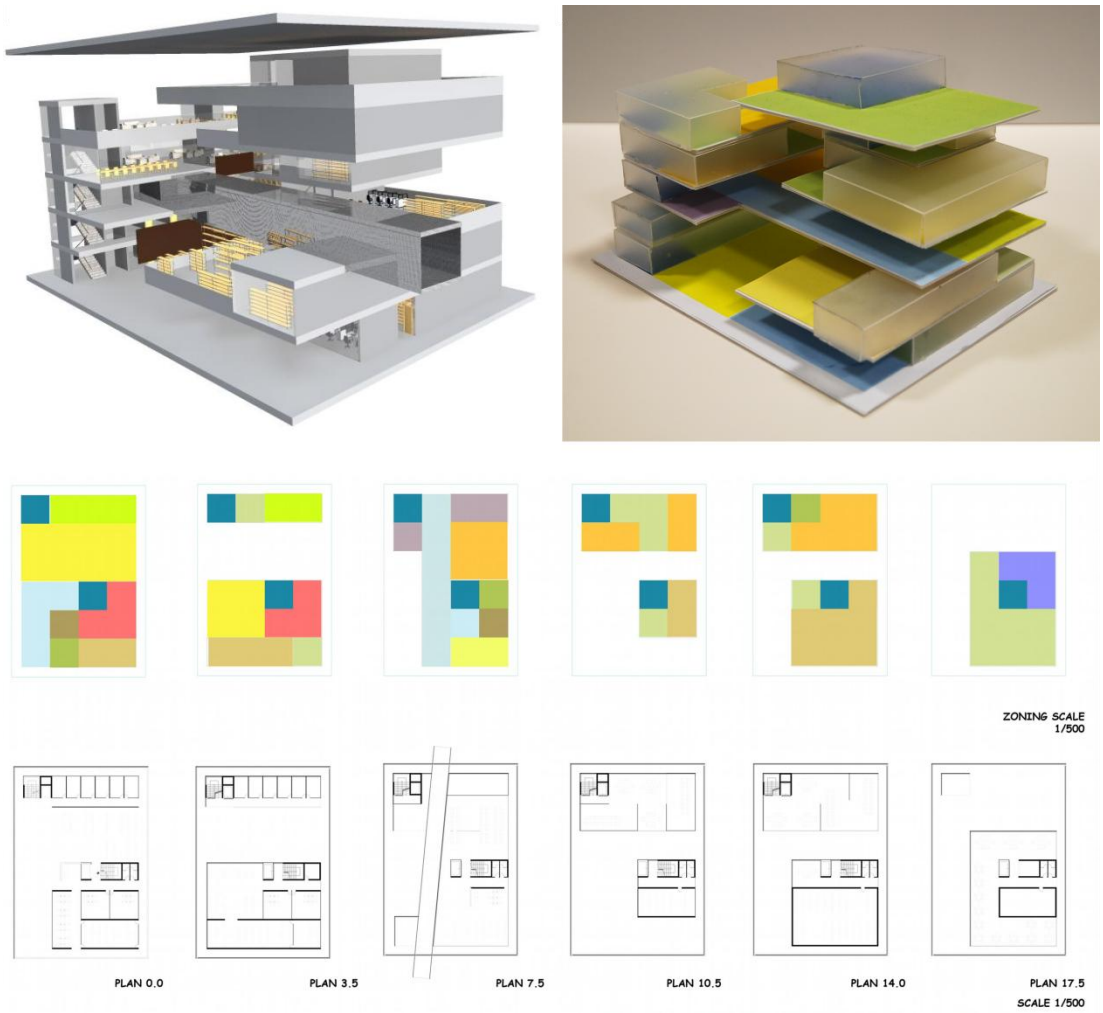


Figure 4.83 : Müge and Ceren, final proposal.

The second example from this workshop is the case of İdil Kantarcı, which was special. İdil was a first year student and had never attended to an architectural design studio. She was a basic design student with absolutely no experience in architectural design because of her school's curriculum. Nevertheless, she was very eager and serious.

In the beginning of the workshop, İdil appeared anxious about not being able to 'exactly' understand the direction she was moving towards, which is a natural response from a first year design student. She quickly lost contact with her teammates and could not connect with other students that were several years senior. She was not confident in her concepts and she had difficulty in sketching out her spatial ideas. Nonetheless, at the end of the day, the only output required by the d_p.layout was a simple floor outline, placed within a site, and this even did not need to be well thought out. İdil managed to carry out this task by cutting out outlines from a piece of cardboard. As can be seen in Figure 4.84, this consisted of three

floors with square outlines and one floor formed by regularly insetting the site borders. There would be a central courtyard, again reflecting the shape of the site borders. This courtyard would include the main entrance from the pedestrian bridge, which would pierce through the building.

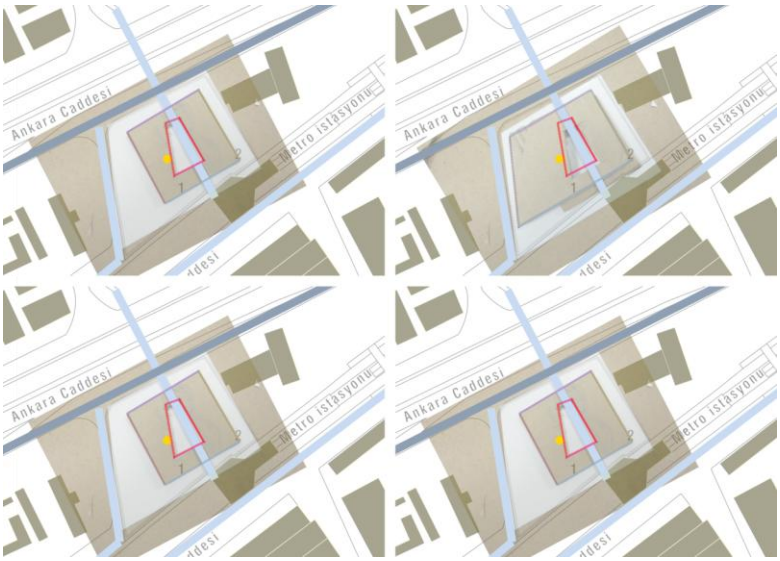


Figure 4.84 : İdil Kantarcı's first d_p.layout trial, as prepared in Inkscape by the tutors.

When İdil brought her first results the next morning (Figure 4.85), it was seen that the last floor was not finished (the draft in the figure developed during the day). A reason was a failure in DU width and area limit settings by the tutor, which could have been far larger with such large floors. Nevertheless, detailed programming and patching studies had been obtained for three floors, from out of almost nothing. Examining these results, it appeared that İdil's total area was far larger than was required, which was far easier to discuss over these detailed scale drafts (consider discussing such issues with a novice that has no understanding of scale).

Figure 4.85 : İdil's first d_p.layout results (DU area limit: 48m^2, DU width limit: 8m)

In Figure 4.85, it can be seen on the second floor (rightmost draft) that there is an abundance of the purple and light green areas, which show activity areas and special sections respectively. This occurrence indicates that the draw at this floor first gave a very large activity hall, which was divided into several DUs until the maximum width limit had been reached. The same occurred with a special section unit, which eventually resulted in a rather unreasonable DU distribution. It appears that the heuristics controlling this occurrence have to be replaced with an approach that probabilistically applies constraints that are specific to DU types, and with respect to the scales and sub-typologies of the proposals. Required information can again be extracted from given buildings.

After obtaining her first results, İdil became able to work on her solid – void model with reference to the draft layouts (Figure 4.86). She had never made building models. This can be observed from her flat spatial models. Nevertheless, through her studies on these drafts, she started to see primary elements of a library layout, circulation system, articulation of spaces, the types of DUs, and so on. Examining d_p.layout outputs amounted to a crash-course on layout generation for İdil and this initiation proved enough to render her productive. She quickly moved towards a better understanding of the spatial organization and started shaping it with her sketches. At the same time, she started to understand how to use d_p.layout, and she started to refine and rationalize her drafts in an attempt to gain control of the process.

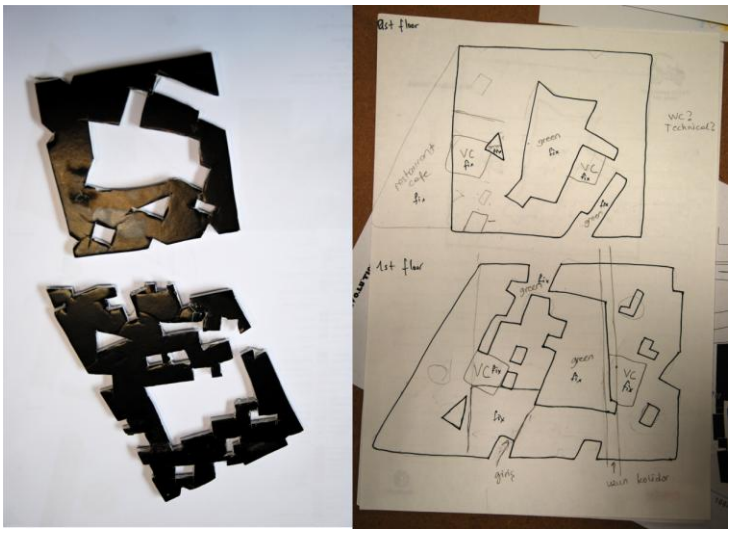


Figure 4.86 : Left: İdil's solid-void exercise for two floors, right: preparation for the second trial.

İdil's second d_p.layout trial can be seen in Figure 4.87. This second trial includes a range of fixed DUs, most importantly the dominant central courtyard that is given as a green space. Several minor green islets, vertical circulations, a large restaurant area, and an entrance hall can be observed in the proposal. Such fixed items make the proposal both more definite (i.e., easier) and more complicated (i.e., more difficult) for d_p.layout.

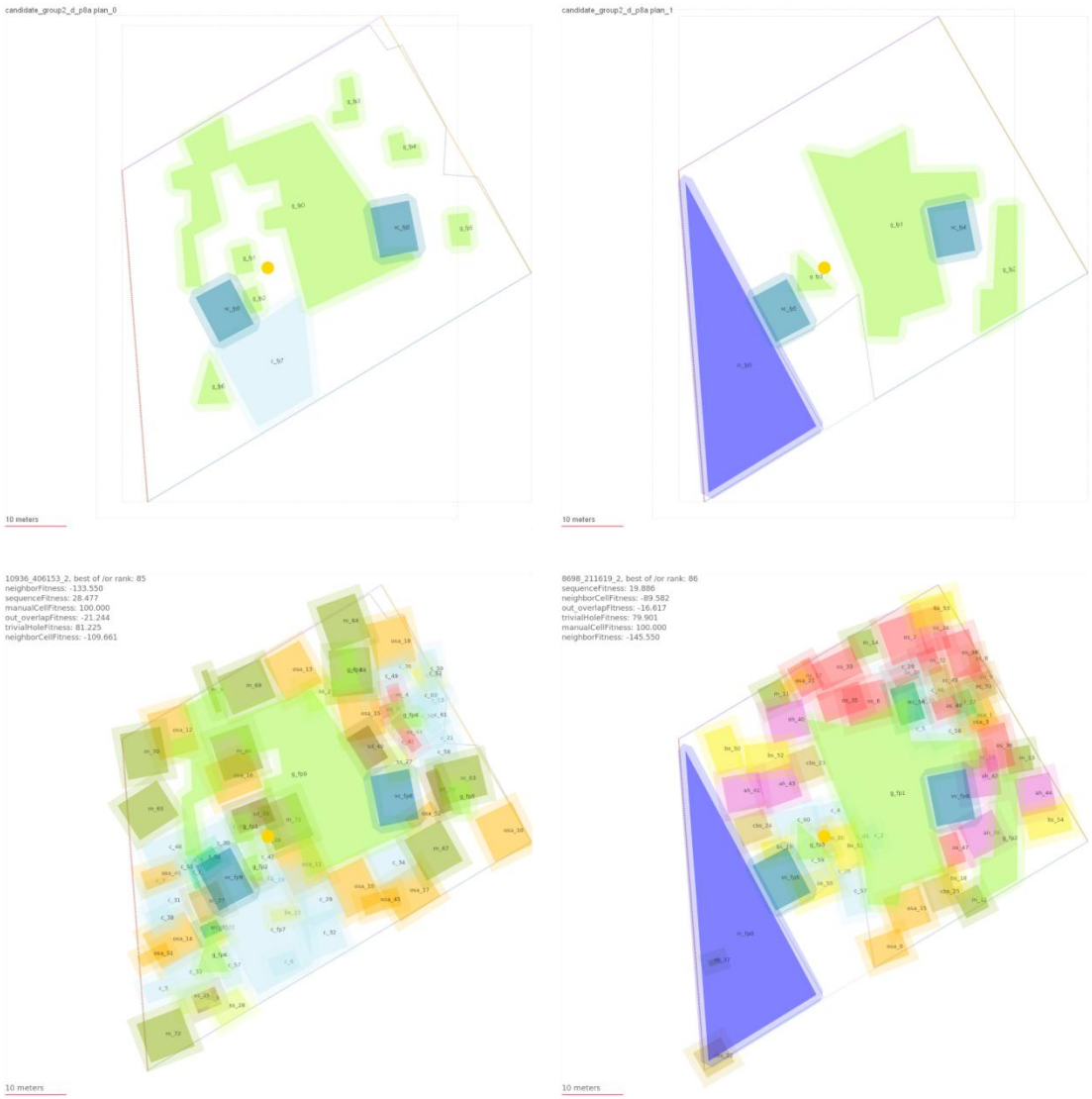


Figure 4.87 : İdil's second trial. Top: initial submission printed after being prepared and parsed. Bottom: outputs of the trial in their original layouts (DU area limit: 50m^2, DU width limit: 8m).

The results display a proliferation of multimedia areas in the ground floor and activity and office spaces in the first floor. Nevertheless, together with the first trial, these drafts offered enough information to İdil that, at this stage she became better able to shape the draft program by herself. When she started working on these drafts, she did not assume these as certain results, and gradually increased her control over the process and refined the functional distribution and shape of her plans (Figure 4.88). At the mean time, she was able to think about vertical separators, visual relations, circulatory system, services, and spatial effects through special elements.

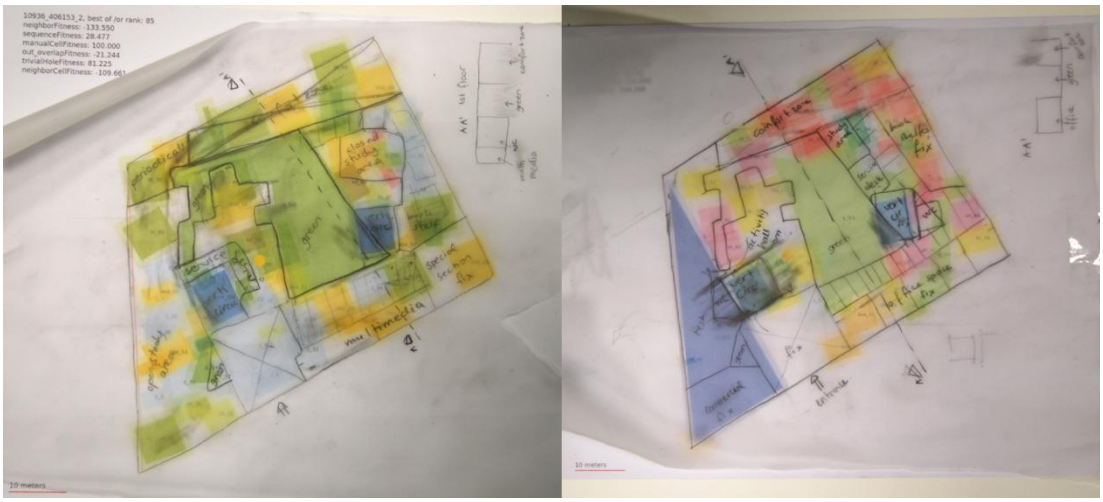


Figure 4.88 : İdil's sketches from the third day.

İdil's final output was quite surprising in its refined articulation of functional areas and intricate, rich, and well-developed interior spaces (Figure 4.89). She collected the multimedia areas on the northern side of the building within a two storey high L-shaped space where a forest of spherical cellular elements would be hanged down to be seen from both the highway and from inside the building through transparent dividers. The central courtyard, which was shaped through a playful cooperation with d_p.layout, presents a sensitive scale and a pleasant spatial articulation. Commercial area, restaurant, offices, activity hall, library areas, and services are all placed and shaped sensitively and in a rational manner. A fascinating result from a first year student's very first venture into architecture. Indeed, better than most of the fourth year student teams.

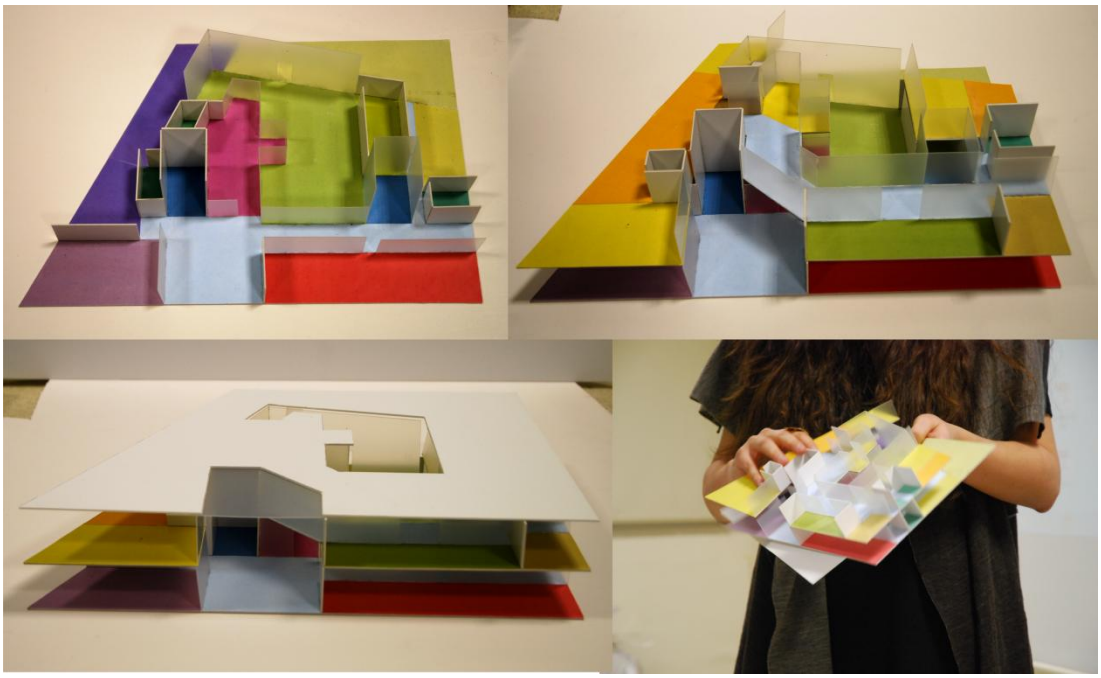


Figure 4.89 : İdil Kantarcı's final product (top-left: ground floor, top-right: first floor over the ground floor).

Surely, İdil was exceptional with her rigor, discipline, and proclivity. Still, it can be claimed that in this case there were facilitating factors added to İdil's exceptional capabilities. We have experienced the rapid development of İdil's capabilities in parallel with the quality of her products and this was well supported by the workshop process. In particular, working with d_p.layout's drafts enabled İdil to practice integral spatial organization while simultaneously learning library functionalities by inspecting and criticizing its proposals. In a way, d_p.layout enabled İdil to structure and express her spatial understanding and productive capabilities. These claims have been corroborated by observations of the second workshop, whose participants were unexceptionally second year students.

In brief, the specific workshop process caused all participants to produce detailed library proposals in five days and the results were mostly satisfactory. All proposals involved interesting aspects and an uninterrupted progression from concepts to spatial organizations. There were technical problems for sure. Some problem definitions had very large floor outlines with high amount of DUs and multiple floors. One night was not enough to evolve these into mature states, which conflicted with the workshop program. In some cases, maximum DU width and area values were not optimally set and some processes crashed due to memory problems and

unidentified code instability. Yet, with all these expected problems, both d_p.layout and the students were flexible enough to adapt to the changing conditions.

4.2.11 Evolutionary Collectivity Mardin workshop

The second Evolutionary Collectivity workshop has been carried out in Mardin Artuklu University, from 17th to 20th May, 2013. The five-day process of the İzmir workshop has been adapted to a period of 2.5 days and tutored together with Ali Paşaoğlu, Figen Işıker, and Asım Divleli from Artuklu University (Figure 4.90). Owing to previous experience, the process became more streamlined. The main components of the process were the same, but emphasis on collectivity was weaker. This did not bring out major differences because the intent for a broader collectivity had not been fulfilled in the first workshop, either. The most important difference between the two workshops was the addition of a short Inkscape course, which enabled the students to prepare their proposals by themselves.

The workshop was part of an architectural design tools course, and all the participants were in their second years. Although with second year students, and although with almost half the available time, the results were more developed than the İzmir workshop. While it is hard to claim that the average level or attendance of the students was better, a potential explanation is that the d_p.layout may be better for beginners.

Figure 4.90 : Photographs from the Evolutionary Collectivity Mardin workshop.

Again, the process started with collective brainstorming sessions (Figure 4.91), and progressed from concept-phrases and sketched out spatial ideas to first models, sections, and floor outlines. Due to the limited time, only one d_p.layout trial has been carried out, which appeared sufficient. The presentations about background information, d_p.layout, and example buildings, and the student presentations were mostly parallel with the first workshop. The second day was a combination of the second and third days of the previous workshop, with the solid – void exercise and the interpretation of the d_p.layout drafts. The workshop ended with the submission of 1/200 models that included draft layouts. In total five teams with eleven students finished the workshop. The given building site for the workshop was well known by the students. It was located amongst scattered buildings with weakly determined borders and with rather vague contextual conditions.

The d_p.layout settings and parameters were mostly the same, with the important exception of the usage of only four objectives (Trivial Hole, Sequence, Neighbor

Cell, and Manual Cell). This meant better average results for each of these four objectives with longer evolution due to slow convergence, which was mostly permitted. Therefore, some floors of the trials could not be finished on time, i.e., in one night. In addition, because the interface was not fail-safe, problems arose in one team's trials, which could not be compensated within the fast progression of the workshop and this team was forced to work over premature results despite the team members' eagerness. Two example processes that have interacted freely and productively with the d_p.layout will be depicted in the following pages.

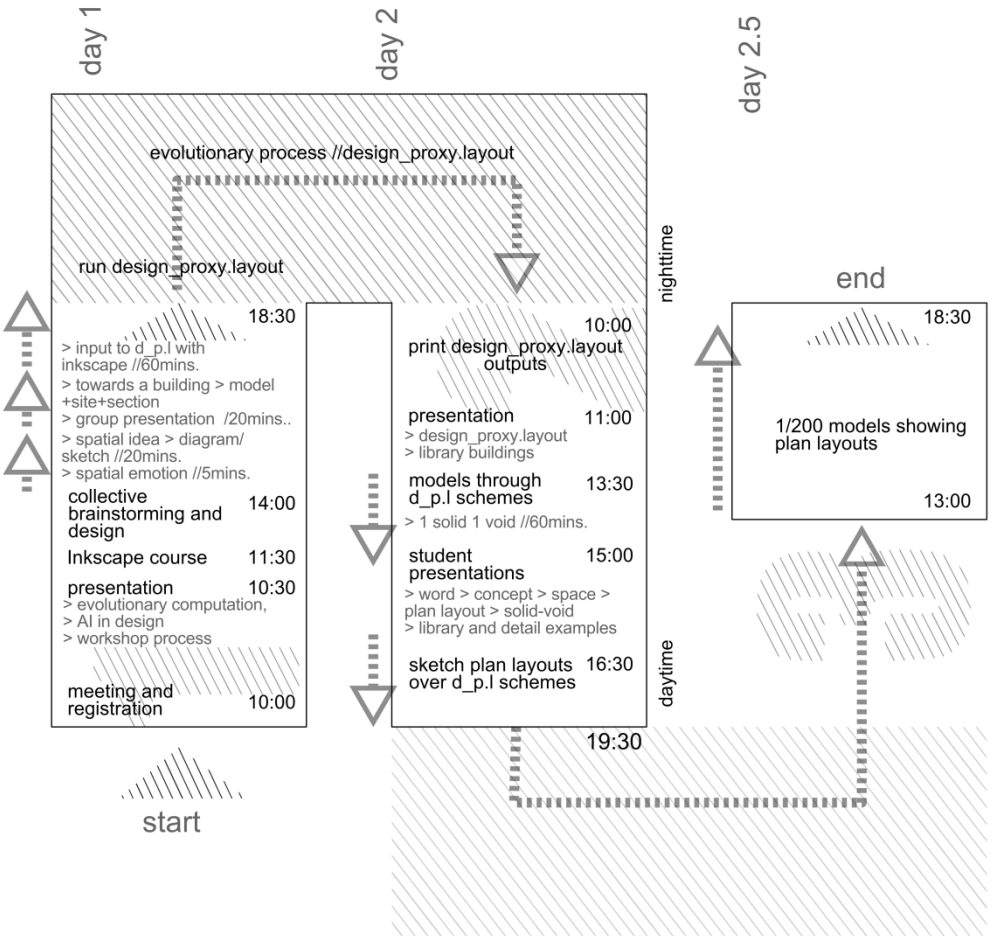


Figure 4.91 : Workshop process for the Evolutionary Collectivity Mardin workshop.

Büşra Tekelioğlu and Merve Çap started their work with the concept of "relaxedness", and tried to reconcile their ideas with contextual conditions in their first model (Figure 4.92), which was dissolved into four masses to adapt to the scale of the scattered neighborhood, while at the same time creating inner streets.

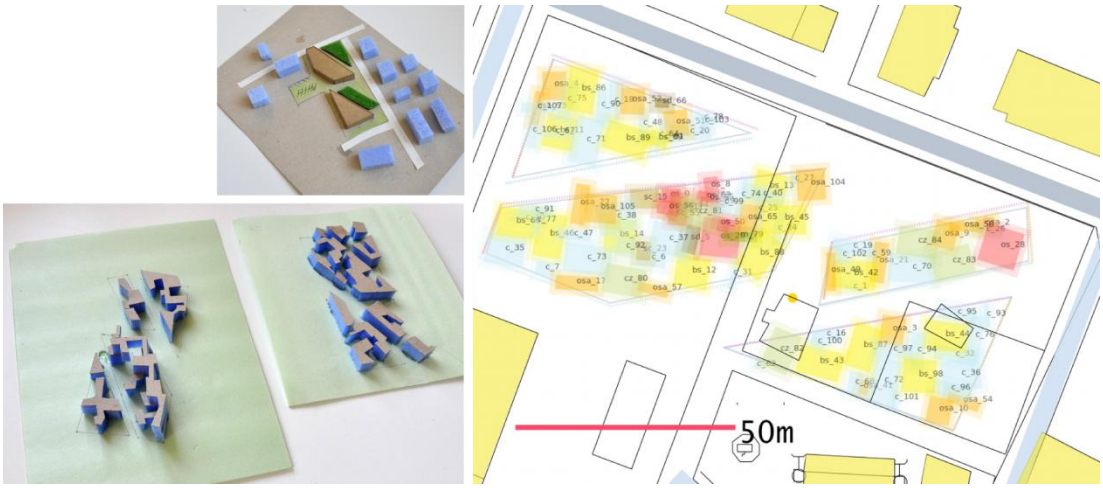


Figure 4.92 : Büşra Tekelioğlu and Merve Çap, top-left: initial massing, right: ground floor of d_p.layout output (DU area limit: 130m^2, DU width limit: 10m), bottom-left: solid-void models.

Sequence articulations and Neighbor Cell groupings can be clearly seen in Figure 4.92 The solid-void studies gave way to the idea of an inner gorge traversing along the length of the masses, which was reflected in the final proposal. The team's progression towards final plans and models can be examined in Figure 4.93. The left column presents functional zoning studies. On the top right, studies on the inner gorge can be seen. On the lower right, these studies can be seen as integrated into schematic drawings.

Figure 4.93 : Second day sketches of Büşra and Merve, from d_p.layout drafts, toward final models.

The final proposal of Büşra and Merve can be seen in Figure 4.94. Adaptation to the context appears as a dominant shaping factor for this project. However, in the end, variety of the in-between spaces, the non-monotonous, fractured interior circulations, visual connections between floors, and created inner perspectives make this project unexpectedly rich in terms of spatial development. This richness in the spatial formation somewhat complicates the functional organization. First of all, the separation of library functions into different masses was not questioned. But this may be due to the characteristics of the historical settlement where the university is located. In Mardin, most historical buildings are organized around or next to courtyards, and circulation from exterior spaces is common. Secondly, both because the students were in their second years, and the workshop had to move forward too quickly, the students tended to follow d_p.layout's drafts in a less questioning manner than the İzmir workshop. In other words, they trusted d_p.layout and used its drafts, and this was the only viable way to finish the workshop in such a short period. Nevertheless, considering the state of development, scale, and placement of the masses, and the building's situation within its neighborhood, the project appears rather convincing; quite open for further development. It is visually pleasing and if was built, it would add a positive quality to its rather undefined neighborhood.

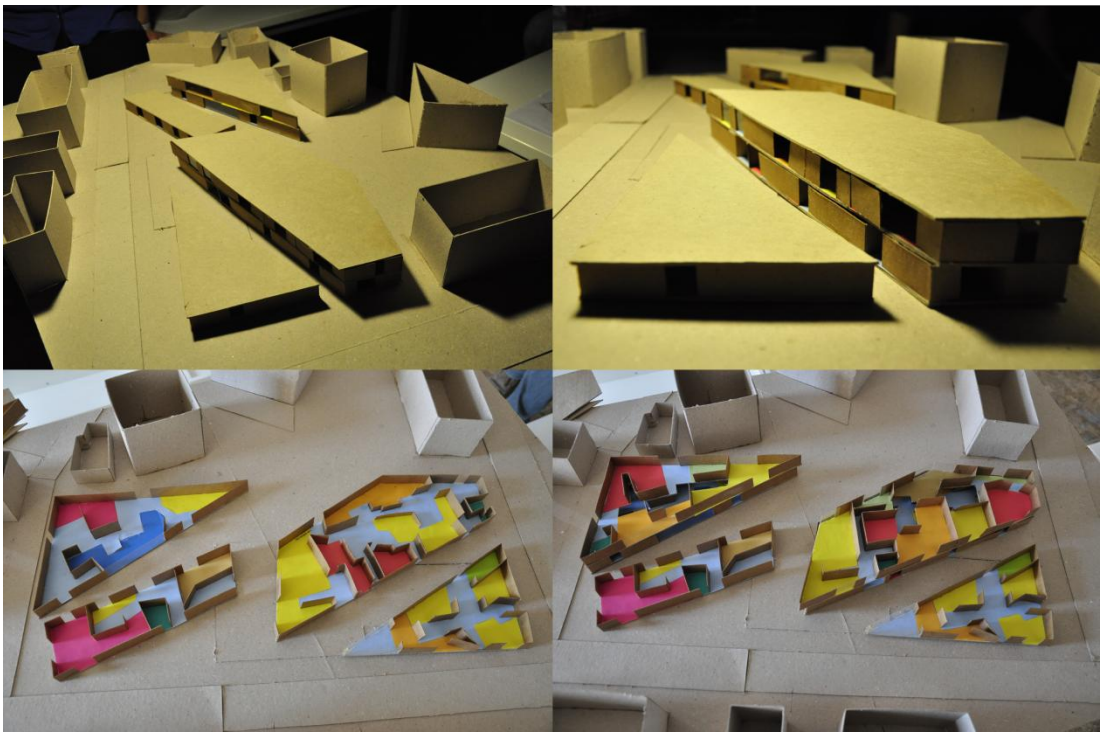


Figure 4.94 : Büşra and Merve, final model (bottom-left: ground floor, bottom-right: first floor over ground floor).

Finally, the project by Merve Bahur, Bahri Özgötürücü, and Melike Kaya was among the most impressive outputs of these two workshops. Their initial concept-word was "depth". Setting out from this word, they decided to emphasize the depth of a building by a gradual passage from brightly illuminated spaces to the darker depths of a building. This was to be realized by a ceremonial procession to the underground through a dominant ramp (Figure 4.95). The outer envelope would be perforated by regular rifts, which were to continue with lesser density on the ground floor slab, so that there would be two light-filters, which would separate spaces in terms of lighting conditions. In their own words:

> "To me, the concept of depth meant thinking and interrogating interior and exterior relations of space. In short, it was asking the question, "Where am I?" And this relationship had to be about the access of light to the space and how it arrives. For me, the depth of the space is its attempt at realizing itself by breaking floor slabs from the sky; at the same time, light has to disappear while proceeding towards that space. For this reason, gliding of the flickering light would make me feel the depth of that space."[25]

[25] Translated from Turkish by the author of the thesis.

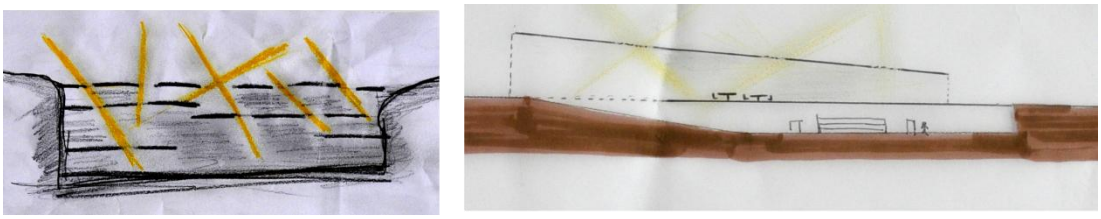


Figure 4.95 : Merve, Bahri, and Melike, conceptual sections.

They have prepared their d_p.layout proposal by fixing a hall that would represent the large entrance ramp on both ground and basement floors (Figure 4.96). Resulting d_p.layout drafts appeared mostly applicable and they became instructive for the team while developing their final proposal, which resulted in dynamic and convenient interiors. Again, the effect of the Sequence (Free University Berlin) and Cell targets (Santa Monica Library) can be clearly observed.

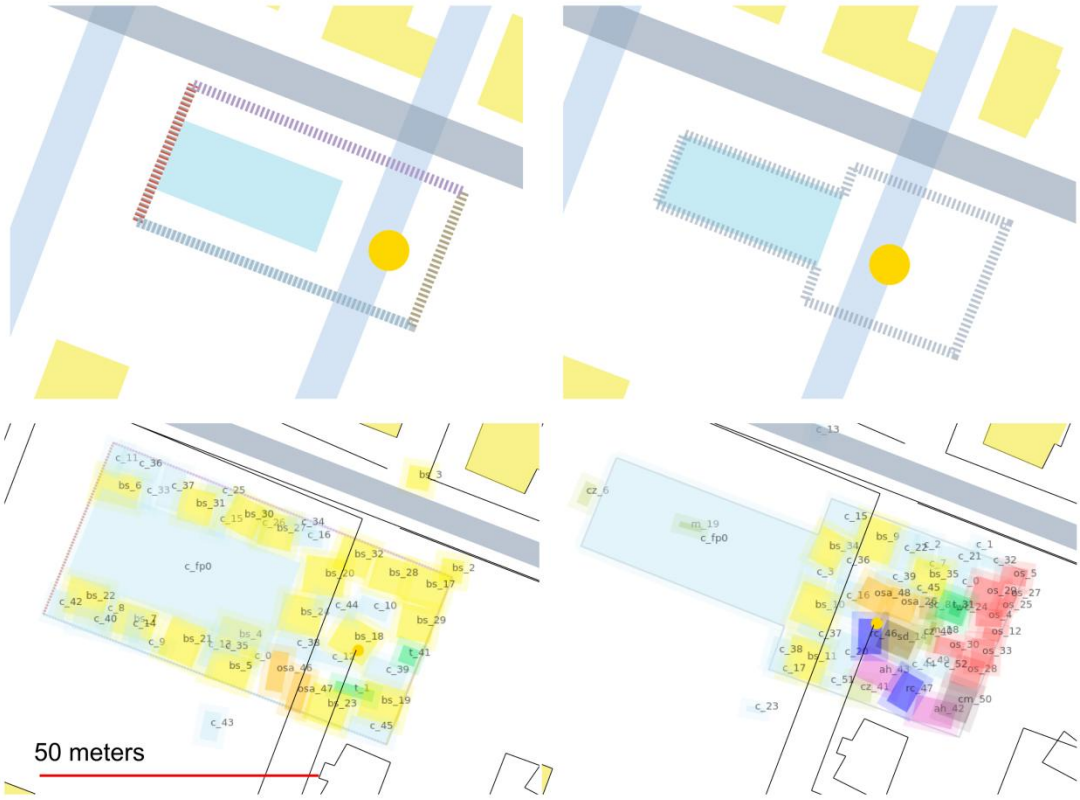


Figure 4.96 : Merve, Bahri, and Melike, d_p.layout preparation and results (DU area limit: 100m^2, DU width limit: 8m).

When they started to work on their solid – void models (Figure 4.97), the team decided to keep the bookshelves on the side aisles within floating bookshelf cubicles.

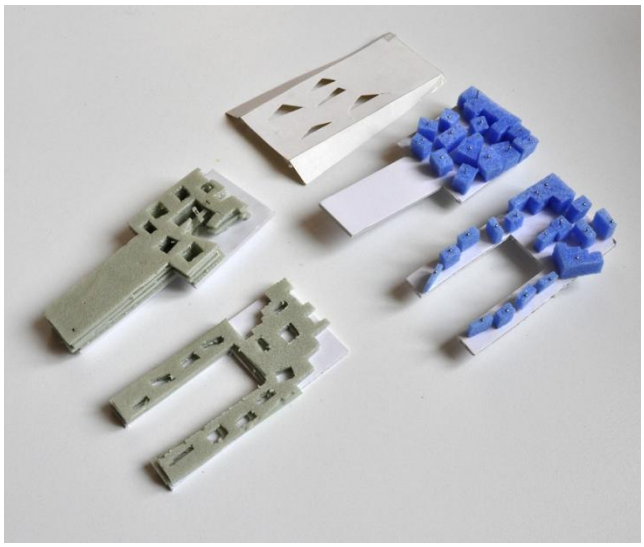


Figure 4.97 : Merve, Bahri, and Melike, solid – void studies and the first envelope proposal.

The team's progression towards their final model can be examined through their sketches in Figure 4.98. The layouts were mostly based on d_p.layout drafts. Over the circulation spaces that were left between DUs, the team started opening floor rifts, which were to be bridged by semi-transparent elements where necessary. The interior spaces were being connected both visually and spatially through both the large ramp and these rifts.

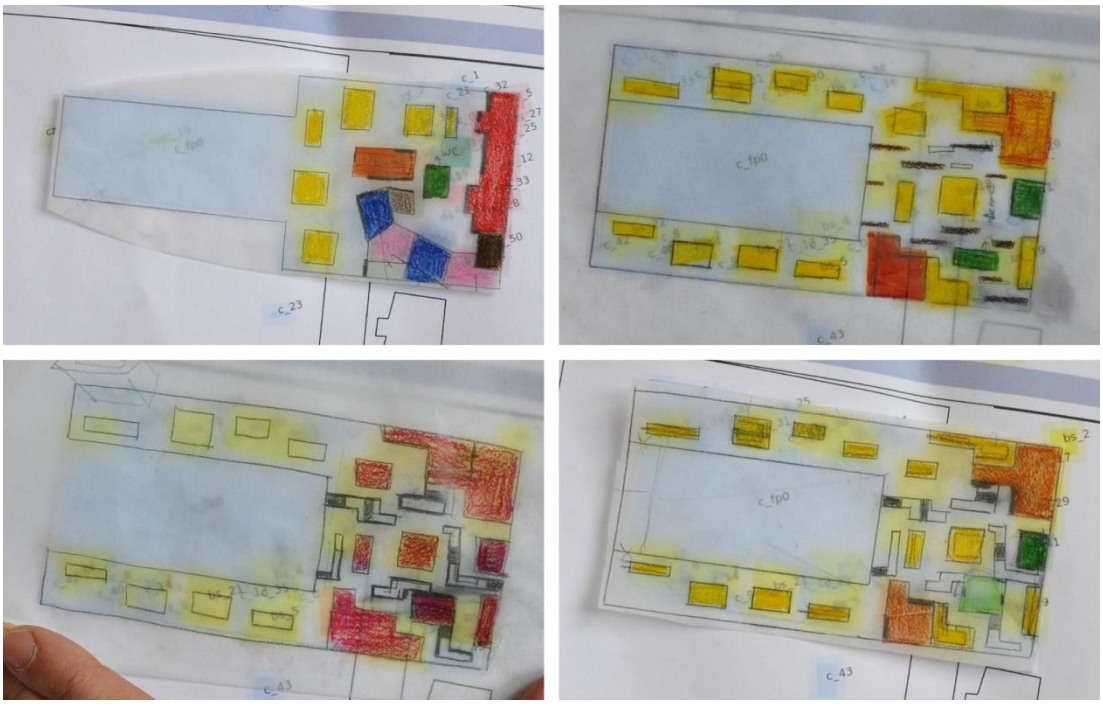


Figure 4.98 : Merve, Bahri, and Melike, sketches toward the final model.

The resulting building is dominated by the huge ramp (Figure 4.99). This could have been precluded or limited by the tutors or the designers in another context as irrational or excessive. However, in the context of this workshop, it was rather costless to move forward and bring the proposal to a level of development before deciding on this issue. In just a few days, the proposal has matured to such a level that it became possible to evaluate the connection between the initial ideas, three-dimensional spatial arrangement, functioning of the building, and contextual relations in combination. The results of this multifaceted evaluation were mostly positive. The building had obtained an iconic quality within its surroundings, the conceptual narrative was consistent and strongly related with the resulting spatial drama, the formation and organization of the rifts were well balanced and unifying, and the dynamism of the functional spaces was reasonably organized. Most importantly, this was just the very beginning and if this were a continuing design process, this proposal would be a very promising initial draft, which already includes many of the crucial architectural aspects to be further discussed, developed, and refined.

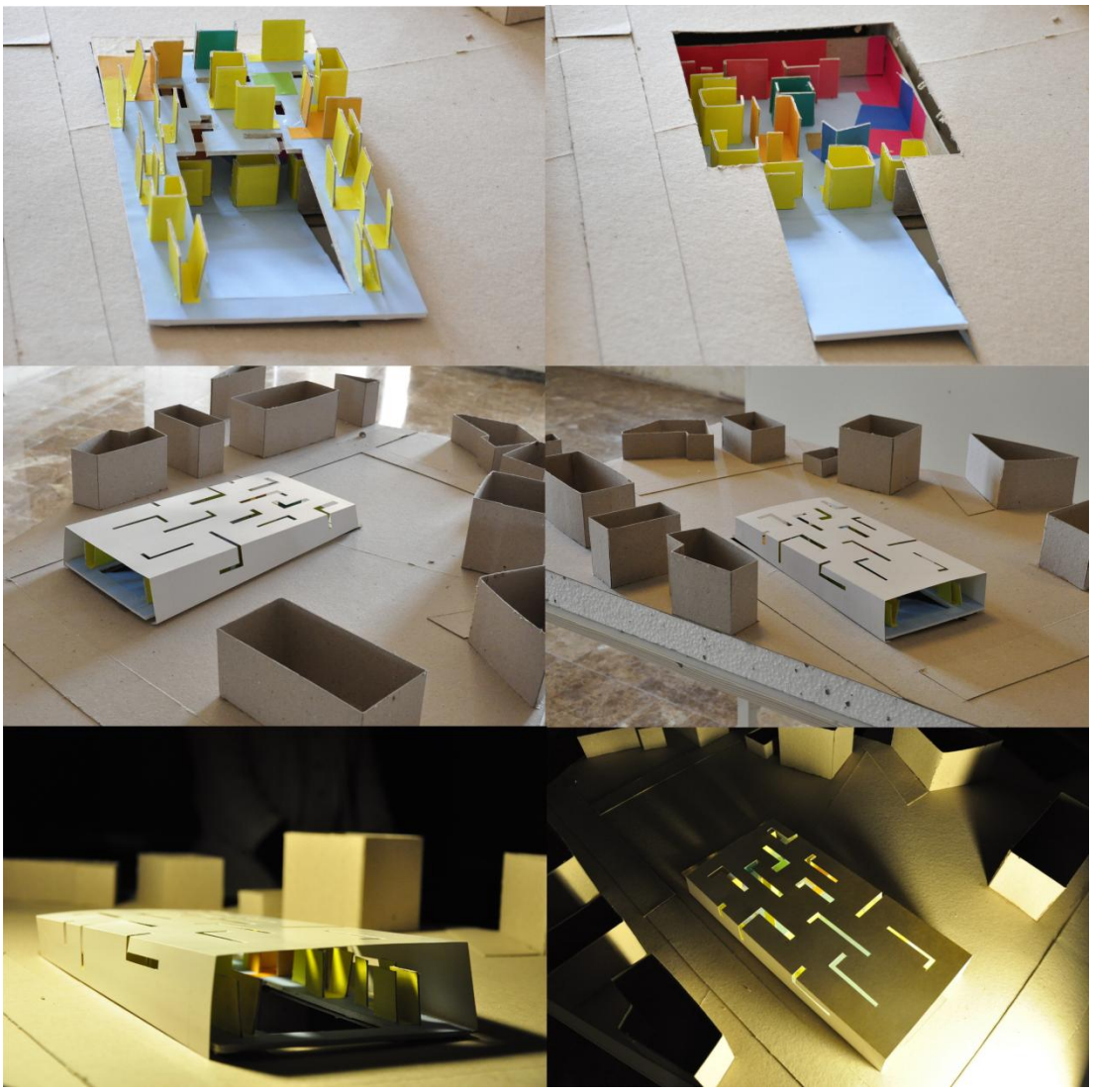


Figure 4.99 : Merve, Bahri, and Melike, final proposal.

Although available time was severely limited, the Mardin workshop was more streamlined and turned out to be more successful than the previous workshop. As the tutors, we have prepared a common site model to be used by all teams and we were able to finish the workshop with an exhibition, yet without a final presentation session. Although the students had to attend to other courses—including their regular design studio courses—they tried to attend to the workshop as much as possible. In the end, the tutors, students, and the staff of the university appeared content with the results. Although through only one trial, the students interacted with the d_p.layout better and were more inclined to exploit its capabilities.

4.2.12 Overall evaluation of the design_proxy.layout

The ultimate heading in the evaluation of d_p.layout is its acceptability within design processes of human designers. The three goals that have to be pursued to be able to accomplish this goal are utility, ease of use, and pleasure; under which a series of evaluation criteria can be listed:

- Utility
    - Technically capable
    - Increases productivity
    - Increases quality
    - Supports creativity
- Ease of use
    - Easily learned
    - Easily used
        - Easy interaction
        - Easy interpretation
    - Easily adopted (within design process)
    - Easily adapted (to a range of cases)
- Pleasure
    - Easy, useful, pleasing, open to playful usage...

Utility can be attained by enabling the user to reap some kind of benefit from a system in terms of productivity, quality, or creativity. To realize these aims, the system has to exhibit a degree of technical capability, as well as offering flexibility to the user, who may develop her own ways to exploit it. If a system is difficult to learn or use, it would only be viable if the ratio of benefit to ease of use is sufficiently high. A more effective strategy would be to design an undemanding system that could easily be incorporated within regular design scenarios. It is obvious that such a system should have an ability to be adapted to a wide variety of cases. It is more difficult to determine evaluative criteria for pleasure; however, it is no less important than the other headings. Therefore, it has to be extracted from overall impressions. In brief, the evaluation procedure is an attempt to assess whether the system responds to these aims and to which degree.

Technical capabilities of the d_p.layout were examined in the previous sections. The observations on the strengths and weaknesses of the system have been mostly corroborated by the workshop applications. The primitive state of the objective functions, small typological libraries, speed and memory issues, and the limited intelligence of the system allows it to produce only rough drafts, or patching studies. However, the results of the d_p.layout are by no means random. Especially in the Mardin workshop, they have largely been followed by the students for simple but convincing layout proposals. The system exhibits difficulties on very large plans (or rather very large DU numbers) and too intricate and narrow outlines, which makes it difficult to fit fixed sized DUs. This limitation can only be overcome by extending the system to enable a variation in DU dimensions. However, such an extension would only be meaningful with the addition of constraining mechanisms. With more refined evaluation procedures, this would amount to the development of the refinement layer for the layout problem, which had been left out in the beginning of the study as was stated previously.

Although by way of subjective impressions, it can be claimed that throughout the two workshops, the d_p.layout system strongly contributed to the productivity, quality, and creativity of most teams; at least the teams that chose to interact with it. This was more evident with the lower year students who depended on it to be able to design, and less with some of the fourth year students, who did not appear eager to compromise their habitual way of design. However, there was no clear separation or groupings in terms of years; rather, generally more or less interested students. A reason for lower eagerness or dependence amongst senior students might be that the d_p.layout's capacities may have been perceived as less competent by these, who were better able to understand the layout task. Nevertheless, there were teams, excellent in their architectural understandings, and very eager to cooperate with the system by freely interpreting its outputs. In short, although mostly positively welcomed, not all students adapted to a workflow with the d_p.layout. This depended on the personalities of the participants, rather than some kind of absolute property of the d_p.layout.

The interface of the system has two constituents. The first is the draft layouts printed on paper. As can be observed through example outputs of d_p.layout, these layout drafts are easy to understand and they leave a wide margin for interpretation. They

can be directly put to use by sketching or tracing over them, which is still the favorite design technique of human designers. The second part of the interface is the input procedure, which is not a separate interface but an existing vector graphics application to be used through template files. This two-part interface may not appear to conform to what is understood by the interface of a computational system, but this choice was deliberate (not only because of practical considerations). The d_p.layout system aims at being painlessly added to the regular design process and only uses the usual tools of designers. It works silently on the background. It is only accessed when needed and through regular interfaces of design. There is no specific interface for d_p.layout.

Inkscape is the popular open source alternative of commercial vector graphic applications and it is both practical and useful for architects. Learning Inkscape is very easy, its workflow is fast, and once learned, it may be used for many other graphic tasks, which combines several benefits. Although it is always possible to develop a specific fail-safe interface to replace Inkscape, this would only mean an added requirement to learn another program. A better future option would be to create a 'fork' of Inkscape dedicated to d_p.layout.

The Inkscape-based problem-definition interface has been used by the tutor in the İzmir workshop, where the students merely submitted their sketch outlines. These hand-made or digital sketches are easily imported into Inkscape. They may also be drawn directly in Inkscape. In the Mardin workshop, after a short tutorial session of around an hour, and with some practice, the students quickly became able to define their outlines and contextual issues.

Learning to use the interface is easy; however, learning to use the system is another matter. In the Mardin workshop, only one trial has been carried out, therefore most intricacies and potentials of the system could not be introduced. However, if the two iterations of İzmir workshop are compared through the examples, it can be seen that the students have easily adapted to the workflow of the system and started to develop an understanding of its potentialities. In such a short time and with just a few trials, at least the more involved students became able to control and manipulate the system.

We have observed that a wide range of problems could be defined within the system's interface. d_p.layout is highly flexible and it is mostly possible to creatively devise ways to define a wide range of layout elements and contextual properties in order to be able to handle new scenarios. This versatility eliminated the need for a constant redefinition of a large number of parameters. There are just a few required parameters, like DU width limits, whose redefinition could have been automated through the same heuristics utilized by the user; i.e., the sizes and dimensions of the layouts.

A convenience of the system is the variability of the control exercised by the user. The amount of control ranges from a more autonomous state (only floor outlines given) to a highly controlled state with elaborate definitions comprising pre-given fixed and mobile DUs, intricate floor shapes, landscape elements, separators, courtyards, directions, etc. The students tended to use these control possibilities after getting used to both the system and the design problems.

These considerations have been based on the subjective interpretations of the author. To be able to present a more balanced opinion, student surveys have been carried out after the workshops. The overall conceptual framework of the workshops was investigating the possibilities of collective design, not only in terms of human-machine collaboration, but also amongst human designers. Although collaboration is a regular aspect of design processes, design offices and teams tend to function hierarchically and most of the times designers carry out the hardest part of the integration work individually, which makes horizontal and participatory design a challenging question. Additionally, design_proxy had set out to function as a collaboration platform and the workshop tutors had an aspiration for developing collective, participatory, and egalitarian process structures for design. Therefore, the first three questions of the survey target the collective aspects of the workshops, without discriminating between the parties of the collaboration (Table 4.2). Question 4 directly assesses d_p.layout. During the presentations tool, assistant, and agent division had been introduced to the students; question 5 measures their opinions on the issue.

Table 4.1 : Evolutionary Collectivity workshop survey.

	(5) A great deal	(4) A lot	(3) Mode rately	(2) A little	(1) Not at all
1. The collective design process during workshop contributed in,					
a. understanding the design problem,	5	4	3	2	1
b. creativity,	5	4	3	2	1
c. productivity,	5	4	3	2	1
d. quality,	5	4	3	2	1
e. saving time,	5	4	3	2	1
f. pleasure (fun).	5	4	3	2	1
2. The collective design process enabled every participant to contribute his or her ideas.	5	4	3	2	1
3. Overall, collective design process was better than conventional, individual design process in terms of,					
a. understanding the design problem,	5	4	3	2	1
b. creativity,	5	4	3	2	1
c. productivity,	5	4	3	2	1
d. quality,	5	4	3	2	1
e. saving time,	5	4	3	2	1
f. pleasure (fun).	5	4	3	2	1
4. During the design process, design_proxy.layout was helpful for,					
a. understanding the design problem,	5	4	3	2	1
b. creativity,	5	4	3	2	1
c. productivity,	5	4	3	2	1
d. quality,	5	4	3	2	1
e. saving time,	5	4	3	2	1
f. pleasure (fun).	5	4	3	2	1
5. design_proxy.layout functions as a,					
a. design tool,	5	4	3	2	1
b. design assistant,	5	4	3	2	1
c. design agent.	5	4	3	2	1
6. What would you add to design_proxy.layout to improve it? What kind of additional functionalities would be nice?					
7. Write your overall impressions: how was the workshop for you? Please add your comments for each of the questions above, especially with a focus on your impressions about design_proxy.layout. (You can use the backside of the form as well.)					

The average results of the surveys can be seen in Figure 4.100. Student impressions on d_p.layout can be examined in the subsections of question 4. Although the students graded d_p.layout above average for all questions, they tended to give lower scores for its contribution in creativity, quality, and pleasure, while tending to think that the system helped more in understanding the problem, increasing productivity, and saving time. These results are correlated for the two workshops, while the scores are lower for the İzmir workshop.

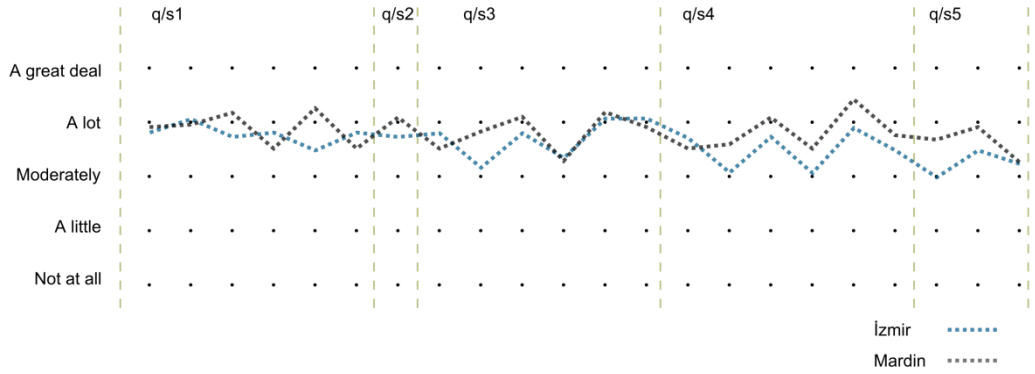


Figure 4.100 : Survey results (mean) for Izmir and Mardin workshops.

In terms of problems, several students in İzmir complained that five days was too long for the workshop (tiresome, hard to focus, sometimes hard to understand), solutions required many refinements, and the process involved cyclic operations, which ended in recurring patterns. The last complaint might be caused by the idiosyncratic Sequence target, which was used for all the trials.

In an interesting comment, a student claimed that the d_p.layout was "somewhat cheating" in education. Working with d_p.layout enabled the beginner students to enter into and develop their problems much faster and in a more detailed manner, which might have caused the students to obtain new capabilities. However, such assisted production could have also hindered the hands-on development of these students. It was clearly observed through the two workshops that the d_p.layout empowers a beginner in architecture. Yet it is not possible to assess, only through these workshops, the medium-term effect of such a tool in the development of a design student.

On the positive side, several students in İzmir thought that the creation of plan variations helped to see different possibilities and the d_p.layout helped in improving

functional layouts, supported thinking differently (hence creatively), helpful for using time effectively, and satisfied as an assistant.

Likewise, several Mardin students claimed that the d_p.layout saved time (quick production and high quality in a short time), helped the production of creative spaces, promoted fast thinking, gave a relaxing assistance, and was fun, very enjoying, exciting (the exciting wait for d_p.layout solutions), and pleasurable overall.

As for potential improvements, the students asked for the consideration of site factors (solar orientation, access, flow, views, noise, pollution), perceptual aspects (sound, light, wind, spatial perception), section analysis in evaluation, and an extension of the target building pool with other typologies (hospital, school, housing, administrative buildings). Some students asked for the addition of layout zoning, partial and detail-oriented operation, coordination of the vertical circulation, and facade design. Faster operation and better quality ("certain") solutions were amongst the demands.

The common occurrence encountered in these different experiences was d_p.layout's acceleration of the process by making it easier to enter into layout generation. While the massing and the contextual and conceptual issues were being considered by the human designers, on the background, the system was generating draft functional programs and was converting these into draft layouts. The second effect of this progression was an easier integration of the different aspects of the process. It is relatively easy for human designers to interpret existing proposals like draft layouts. Compare this with a blank slate that has to be somehow structured, shaped, or filled; which can be seen as an unstructured situation, an open problem with no structure, with no path to move forward. On the other hand, whenever human designers face draft proposals, it becomes easier for them to move forward by interpreting and refining these; the problem gains structure and the designers are set on a path. This makes the process both easier and more pleasurable. Therefore, an assistant who is capable of bringing forward rough proposals, although inferior, would have a utility within the design process. While the human designer is a natural draft modifier or refiner, the d_p.layout is a draft producer: First, its outputs are drafts. Secondly, it assists in the production of fast draft proposals.

As can be seen from the examples, the d_p.layout can be put to use in the very beginning of a design process where the problem is rather vague and the information is sparse (not yet collected by the designer). At this point, it can assume the burden of analyzing existing buildings. Using this extracted information, it generates flexible drafts that are open to development and enables its user to jump over these initial steps directly into design refinement with a less painful progression, which allows the determination of the problem in many aspects simultaneously—spanning several layers of conceptual design—through fast but detailed probes into the problem (i.e., solution alternatives). Throughout the workshops, the d_p.layout helped the teams to give initial structure to their design processes and to develop initial proposals. At the end of the workshops, these proposals were at such a stage that they were allowing the students to discuss many aspects of their problems and proposals in an architectural manner.

Although these workshops do not assess the potential professional applications of the d_p.layout, it appears that such a workflow could have been employed to quickly develop several initial ideas, just like an in-office competition. In this sense, the d_p.layout lengthens the process of brainstorming into more detailed design layers. Although the d_p.layout is not highly intelligent, in the hands of its smart users, it turns into an idea generator. If it could enter design practice from this point, this could open the way for its further development, towards a higher level of intelligence.

4.3 Conclusion

In this chapter, the design_proxy approach and the Interleaved EA were applied to two design tasks. In these applications, the multi-objective evolution approach, procedures for adaptivity, and the evolutionary process were attributed to the Interleaved EA, while the aspects of the specific application, i.e., task definition, representation, initiation, evaluation, variation methods, and the interface were collected under a specific instance of the design_proxy, i.e., d_p.graphics and d_p.layout.

The d_p.graphics was a toy application, used to demonstrate and test the naive version of the Interleaved EA. However, the d_p.layout is a versatile and robust layout design assistant and can be used by architects in their design processes. Figure

4.101 shows how specific techniques that were adapted and developed for the system correspond to the tenets of the design_proxy approach. First, the system uses a relaxed problem definition (patching drafts) and a flexible layout representation that permits the overlapping of DUs and boundaries. Secondly, the interaction with the program (both input and output) is carried out through intuitive 2D graphics. Thirdly, the system carries out functional evaluations by measuring the similarity of a proposal to existing target layouts. Finally, it utilizes the rank-based version of the Interleaved EA. Functioning in an integrated manner, these properties of the system make it a practicable and enjoying design assistant, which was demonstrated through the two workshop cases.

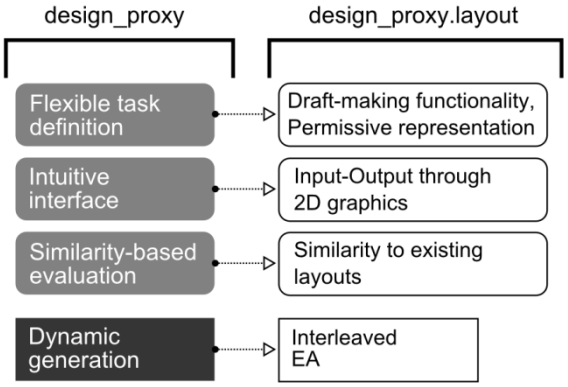


Figure 4.101 : design_proxy approach and its application for design_proxy.layout.

Before demonstrating the d_p.layout system, the chapter located the architectural layout problem within the broader problem of the conceptual design of buildings. This resulted in the proposal of a conceptual framework for artificial architectural design assistants, i.e., the Architectural Stem Cells (ASC) Framework. The ASC Framework proposes a dynamic and multi-layered method for combining a set of design assistants for larger tasks in architectural design.

The first component of the framework is a layer-based task decomposition approach inspired by Brooks's (1999) parallel task decomposition for the subsumption architecture (Figure 4.102). The idea is applied to the sub-tasks within architectural design to obtain a dynamic parallelization of these tasks (Figure 4.28), which are conceived as the layers of the same development process.

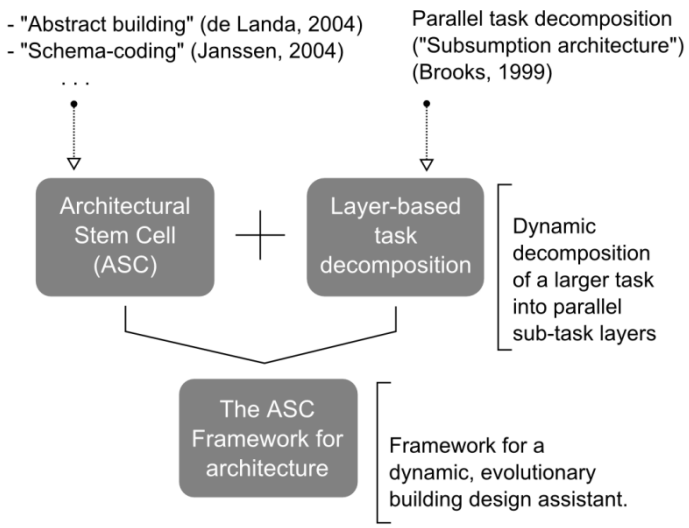


Figure 4.102 : Basic constitution of the ASC Framework.

The second component of the ASC Framework is a conception for abstract building development specifications, i.e., Architectural Stem Cells (Figure 4.102). The ASCs are required for re-integrating the separated task layers of the architectural problem through solution-based development (Figure 4.30). An ASC can be conceived as a semantically marked geometric structure, which contains the information that specifies the possibilities and constraints for how an abstract building may develop from an undetailed stage to a fully developed building draft.

The ASC idea has several precursors in the literature. While a conception for an "abstract building" has been proposed by de Landa (2004), this was an undetailed proposal to conserve topological properties while changing the dimensioning. As was illustrated through the manually developed ASC examples (Figure 4.30), in contrast to de Landa's proposals, an ASC approach should permit adding, changing, displacing, and subtracting the properties and arrangements of a developing instance wherever necessary, as is the case with real design processes. Another direct precursor for the ASC idea is Janssen's (2004) "schema coding" stage for evolutionary design, which would encapsulate all the rules and representations for the linear unfolding of a building design through evolution. There are several differences between schema-coding and ASCs, and these may have important consequences. The ASC idea involves the configuration of each ASC in a way that it would conform to the task layers of an overall problem. Different task modules can operate on the same ASC instance throughout its development process. These task modules correspond to the layers of the decomposition scheme. Thus, an ASC’s

development process may be defined and controlled by a layer-based decomposition approach, which, by relegating each of the tasks to a separate module, may make it possible to sidestep the immense complexity of the overall task. As another consequence, because these task layers are conceived as parallel instead of consecutive, the overall system may exhibit a dynamic, non-linear character, where each of the layers may have simultaneous effects on the others.

As can be seen through the d_p.layout system, evaluation through similarity opens a path towards open-ended evolutionary systems that can learn, and creates a potential for contextual evaluations. Together with such methods, the ASC Framework may be showing the path towards dynamic, open-ended, and contextual architectural design assistants. The ASC Framework brings together many of the issues of this thesis in a structured manner and puts forward a possibly long-running research agenda.

5. CONCLUSIONS AND FUTURE RECOMMENDATIONS

This study investigated the requirements for a draft making design assistant and developed approaches, techniques, and computational tools for the realization of such an assistant. For this aim, the study carried out a broad investigation into the main problems, challenges, and requirements towards such assistants, through both theoretical investigations and practical applications. The main technology explored for this aim was Evolutionary Computation (EC), and the target design domain was architecture. Thus, the two connected research questions of the study concerned, first, the investigation of the ways to develop an architectural design assistant, and secondly, the utilization of EC for the development of such assistants. Besides theoretical and interpretative outcomes, the basic outputs of the thesis are[26]:

- The "design_proxy" approach (d_p):

  An integrated approach for draft making design assistants. It is an outcome of both theoretical examinations and experimental applications and proposes an integration of (1) flexible and relaxed task definitions and representations (instead of strict formalisms), (2) intuitive interfaces that make use of usual design media, (3) evaluation of solution proposals through their similarity to given examples, and (4) a dynamic evolutionary approach for solution generation. The design_proxy approach is expected to be useful for AD researchers that aim at developing practical design assistants.

- The "Interleaved Evolutionary Algorithm" (IEA, or Interleaved EA):

  A novel evolutionary algorithm, proposed and used as the underlying generative mechanism of design_proxy-based design assistants. The Interleaved EA is a dynamic, adaptive, and multi-objective EA, in which one of the objectives leads the evolution until its fitness progression stagnates. It leads the evolution in the sense that the settings and fitness values of this

[26] Also see Figure 1.2. Note that the design_proxy.graphics application, although listed as one of the research instruments, is not considered among the final outputs.

objective is used for most evolutionary decisions. In this way, the Interleaved EA enables the use of different settings and operators for each of the objectives within an overall task, which would be the same for all objectives in a regular multi-objective EA. This property gives the algorithm a modular structure, which offers an improvable method for the utilization of domain-specific knowledge for each sub-task, i.e., objective. As a specific EA variant, it can be used by Evolutionary Computation (EC) researchers and by practitioners who employ EC for their tasks.

- The "Architectural Stem Cells" (ASC) Framework:

  A conceptual framework for artificial architectural design assistants. It proposes a dynamic and multi-layered method for combining a set of design assistants for larger tasks in architectural design. The first component of the framework is a layer-based, parallel task decomposition approach, which aims at obtaining a dynamic parallelization of sub-tasks within a more complicated problem. The second component of the framework is a conception for the development mechanisms for building drafts, i.e., Architectural Stem Cells (ASC). An ASC can be conceived as a semantically marked geometric structure, which contains the information that specifies the possibilities and constraints for how an abstract building may develop from an undetailed stage to a fully developed building draft. ASCs are required for re-integrating the separated task layers of an architectural problem through solution-based development. The ASC Framework puts forward a research agenda and it is presented to the AD researchers in architecture.

- The "design_proxy.layout" (d_p.layout) application:

  An architectural layout design assistant based on the design_proxy approach and the IEA. The system uses a relaxed problem definition (producing draft layouts) and a flexible layout representation that permits the overlapping of design units and boundaries. User interaction with the system is carried out through intuitive 2D graphics and the functional evaluations are performed by measuring the similarity of a proposal to existing layouts. Functioning in an integrated manner, these properties make the system a practicable and enjoying design assistant, which was demonstrated through two workshop

cases. The d_p.layout is a versatile and robust layout design assistant that can be used by architects in their design processes.

In the following sections, the rationale behind each output, the expected practical benefits, and the targeted user groups will be presented in more detail, and the potentials and weaknesses for each output will be discussed together with future recommendations. Additional theoretical and interpretative outputs, overall considerations on future research paths, and recommendations will be discussed in the final section.

5.1 design_proxy Approach

The design_proxy (d_p) is an integrated approach for draft making design assistants (Section 2.6). It is an outcome of both theoretical examinations and experimental applications. It proposes an integration of (1) flexible and relaxed task definitions and representations, (2) intuitive interfaces that make use of usual design media, (3) evaluation of solution proposals through their similarity to given examples, and (4) a dynamic evolutionary approach for solution generation. As was shown with the two applications of the approach (d_p.graphics and d_p.layout; Chapter 4), it can be used for different design tasks. Its most concrete practical instance is the design_proxy.layout assistant, whose utility has been shown in educational settings (Sections 4.2.9-12).

As an integrated set of loose guidelines, the design_proxy approach has been proposed as a reusable strategy for the use of Computational Design researchers. Its central tenet, the similarity-based evaluation approach, which makes use of the information that resides in existing examples, together with the demand for a dynamic generation process, offer paths that are indefinitely open to further development and may inform and inspire further studies. In particular, the approach can be used by AD researchers that aim at developing practical design assistants.

5.2 Interleaved Evolutionary Algorithm

The thesis proposed and studied EC as the underlying generative technology for draft making design assistants, as the operating level of the basic EC mechanism appears compatible with the basic available level of AI for design. For this aim, together with

an overall introduction to EC in design, the thesis proposed and tested a novel evolutionary algorithm. The Interleaved EA is a dynamic, adaptive, and multi-objective EA, in which one of the objectives leads the evolution until its fitness progression stagnates. In this way, it enables the use of different settings and operators for each of the objectives within an overall task (Section 3.7). As such, it is a concrete, reusable tool. It may be used and tested by all EC researchers and practitioners who use EC in tackling their problems.

Besides being a readily usable tool, the Interleaved EA aims at setting forth a research path towards a higher level of dynamicism by offering slots for gradually incorporating new intelligent technologies for contextual state recognition and design specific intelligence.

In EC, the complexity of a task increases with the number of simultaneous objectives, and this complexity impairs the overall success of the results. Unfortunately, a complicated task like designing a building may involve a huge number of objectives and requirements. Intelligent executive and evaluative mechanisms may improve success; however, the real remedy appears as dynamically breaking down and re-integrating a complicated process into less complicated sub-problems on the run. This dynamism should be the next level for the computational evolution of design. The main requirement for this dynamic structuring is an intelligence capable of appreciating the design context in order to inform the focusing action. This structuring also requires a corresponding complexity in the available EAs. This is why several schemes for complex and hierarchical EAs have been presented in Chapter 3.

The Interleaved EA follows this vision for a dynamic EA, and has been envisaged as a step towards such an aim. It focuses dynamically on sub-problems, so that at each stage only one of the objectives is dominant with its specific operators and settings. The naive version of the Interleaved EA consists of a series of single-objective evolutionary runs, each constituting an evolutionary goal module, operating over the transformation of the same species of solution proposals according to the feedback gained from the course of an evolutionary process. This modular structure of the Interleaved EA was thought as reminiscent of how a human designer works, who focuses temporarily on a limited set of aspects, and then, after improving on that set, moves forward to improve another group of aspects, which was possibly degraded by

the first occupation. In future studies, these modules might be engineered to correspond to the basic actions of design processes, which are being postulated by protocol studies.

It has been shown by Goldschmidt (2006) that, on a micro level, human designers employ rationales for directing their actions. These rationales appear as scattered throughout a design process, although this does not mean that each action is coupled with an atomic rationale. Nonetheless, it would be expected that, whenever the Interleaved EA temporarily leaves a goal module and goes on to handle another one, this would be related to a rationale. In general, EC requires the constant monitoring of the migration of a species within a fitness geography, in terms of each objective, which provides a convenient mechanism for establishing rationales. Whenever the fitness progression is slow or negative, a change in the parameters may be beneficial. Accordingly, the Interleaved EA acquires its rationales through a constant monitoring of fitness progressions. Additionally, because each objective is being monitored separately, the new move can be related to a specific goal module if desired. This does not mean that new actions could be guided merely over an analysis of fitness progression. In practice, process and product variations are mostly randomized, and the choice is deferred, to be determined by the success or failure of the outcomes.

As would be expected, the naive version of the Interleaved EA was not working for conflicting objectives. The fitness graphs were expressing chaotic wave forms, meandering up and down due to an objective's being active or not. Each objective was working for the detriment of the others, resulting in no considerable rise in the overall fitness. Therefore, an additional mechanism was required, which would prioritize the moves that would not be detrimental for other objectives. Consequently, a rank-based approach is adopted for the population selection phases of the Interleaved EA. This approach enables the simultaneous improvement of all objectives while retaining the separation of the operators and settings of the objectives.

The Interleaved EA has been tested for two different problem areas, i.e., graphical arrangements and architectural layout design, which are both 2-dimensional domains. These two task settings were deliberately similar—for the sake of the continuity of the research efforts—yet, it would also be interesting to tackle a 3-dimensional

design task. Additionally, the task remains, to compare the performance and specificities of the rank-based Interleaved EA with other multi-objective EAs, which is left for future studies.

5.3 design_proxy.layout

The d_p.layout is a versatile architectural layout design assistant based on the design_proxy approach and the Interleaved EA. It can be used by practicing architects in their regular design processes. Thus, it is not only a case study, but also a concrete output on its own. The basic capability of the d_p.layout is generating drafts for architectural layouts. Through these drafts, it functions as both an analysis tool and a proposal development assistant.

To be able to develop itself in a mutualistic environment, a design assistant has to start its career as soon as possible. It may start its functioning as a draft maker, which makes EC a good option for design intelligence studies, for EC is capable of yielding draft results for design. Although these drafts approach nowhere near the quality and sensitivity of a human designer's proposals, they may still ease difficult tasks for human agents by functioning as means to contemplate on alternatives, or as intermediary points to be revised and improved. If artificial systems, using such technologies, are admitted within human design environments, researchers will gain an opportunity to develop better systems with the cooperation of human designers. Future studies should be concerned with developing methods to learn the more effective techniques while the assistants are being employed for design.

The graphics-based interface of the d_p.layout provides a fast interface for transferring information in an easily interpretable form. Working on graphics and over printed images is intuitive and natural for designers. Still, it may be questioned, whether this is a proper interface; since it does not match with the usual prototype of computer application interfaces. Indeed, it is trivial to develop such an interface by coding a custom Inkscape fork. However, considering the interface, the real potentials rather lie in an extension of the types of the outputs. The current d_p.layout system delivers only patching studies, i.e., schematic plans, yet it would be rather straightforward to express the same information in terms of sections, 3D models, renderings, and animations.

It should be remembered that the choice was deliberate in keeping with the regular and available tools and media of designers. AD research is directed towards autonomous agents, placed within architectural production spaces, as with the other agents. Therefore, what is lacking with the d_p.layout is not a dedicated computer interface or other types of representations on screen; the main potential rather lies with 3D media. With recent developments in affordable 3D technologies, there appears a potential for artificial agents to be interfaced by 3D controllers, vision systems, and printers. This would be very much in line with the vision of Negroponte, hence with AD, too. Although natural language processing would be lacking in the beginning, at least the formal manipulating operations could be readily incorporated into such an environment. Such developments could be justified either with the intelligent support given by the system, or with these tools' utility in regular design scenarios led by human designers. The latter appears as the more probable scenario for the near future. Thus, AD research should keep up with the developments of both traditional and novel design tools of humans, and continue calibrating itself to them, instead of trying to express itself with specific interfaces. After all, even with its modest tools, this study has shown potentials of such natural interfacing between human and artificial agents.

Similarity-based evaluation appears as a significant proposition of this thesis. Although it is not absolutely novel, as is traced in the dedicated literature reviews, the design_proxy applications have nevertheless tried to broaden both practical and philosophical horizons for this approach. First of all, basing evaluations on a pool of existing designs or quickly delivered graphical specifications enables EC to become an open-ended system. This way, apparently strict problem definitions have been kept open to fast updating, which should be compared to rather static approaches like adjacency matrices, lists of constraints, and fixed objective functions. The resulting assistants are able to be quickly adapted to a range of problems, none of which is strictly implied within the initial framework or task definition. Moreover, the same approach can be extended towards adjacency matrices and design constraints, by extracting these from existing buildings on the run, which would convert them into dynamic approaches. This is exemplified with the "Adjacency" versions of the objectives in the d_p.layout application.

However, although able to function to some extent, the implemented evaluation procedures are rather primitive, at least when compared to human capabilities. Other techniques are required to develop these further. For example, the d_p.layout could be a better interpreter by using statistical methods and datamining, and it could be better informed by using machine-learning techniques. These avenues should definitely be followed in future research. Through these kinds of techniques, different spatial relationships may also be incorporated into existing evaluations, such as orientation, access, neighboring buildings, views, etc., whose information has already been included in the target examples. In addition, distinctive features for new and more sensitive comparisons may be detected with datamining techniques.

The development of evaluation techniques over target buildings is only one aspect of the problem. A second issue is the constitution and utilization of the example pools from which the targets are selected. As the d_p.layout is dependent on the distinction of a series of architectural typologies, a large example pool with different typologies becomes a straightforward requirement (currently only a small set of library buildings are available in the target pool). However, the determination of typologies is not straightforward in all cases. The problem extends towards sub-typologies, hybrid cases, and situations where no definite typology exists. These are open questions for the d_p.layout. Hybrid methods that would comprise manual labeling, datamining, and clustering analysis may be investigated for potential solutions. Even when a definite typology is detected, there might not be enough information for a specific DU type in a target building. This was the case with commercial areas in the d_p.layout applications, which results in rather random placements. This clearly shows a requirement for methods that would enable the collective usage of a series of targets together, as is the case with the distribution of DUs to floors.

The current mechanism for the distribution of the DUs to separate floors has a basic drawback; once the DUs are distributed to the floors, the lists are fixed. This could be overcome by dynamic DU lists or through a parallel evolution of the floors. The latter would allow the swapping of DUs between floors as well as a vertical coordination of the functional zones, thus it is highly desirable.

A refinement functionality was not demanded for the draft level outputs. However, as the current level of the system has been set only as the starting point, a gradual improvement of the system towards more refined proposals is highly desirable; most

probably with the addition of refinement task layers. There are several requirements for such improvements. First, the variation operators have to be modified to enable the adjustment of dimensions of DUs. Secondly, new evaluation procedures are required to appreciate the relative merit of more refined instances. After refinement becomes possible, it can be incorporated into the problem definition. An alternative strategy could be to use different representations for refinement tasks, which would be interlinked with the current polygon representation. For example, schematic line drawings could be drawn over a point-cloud version of the current patching polygons. Although not trivial, this appears as a viable approach. Ideally, the refinement layer should have an option for simultaneous evolution with the other layers. A straightforward approach would be to use transformations between different representations for a communication between different task layers, from polygons to point clouds to line drawings and vice versa.

One of the issues that are considered important during the development process of the d_p.layout was versatility, which is connected to both ease of use and the adaptability of the system. These two aspects overlap in parameter adjustment operations, which were attempted to be eliminated by adaptive parameter control, which has not been successful for the layout case. This remains an open problem because in some scenarios performance of the system may become critical. Future studies should continue the search for better parameter-control approaches, for which the most promising path is the self-adaptation approaches developed within the Evolution Strategies lineage of EC.

The usage cases of the d_p.layout were situated in educational contexts, where the users were students of varying levels. As we propose it as a tool for practicing architects as well, the task remains to employ it in real world design situations.

5.4 Architectural Stem Cells Framework

The full potential of the design_proxy approach can only be realized if the approach is used for different layers of a complicated design task within a dynamic and parallel development setting. However, this appears as a huge task that far surpasses the limits of a single thesis study. Nevertheless, the Architectural Stem Cells (ASC) Framework is developed to examine potential paths for such a dynamic and multi-layered system, which would combine a set of design assistants for larger tasks in

architectural design. Layer-based task decomposition and ASC approaches can also be understood as a unified answer to the question "how may creative, conceptual architectural design be structured to constitute a basis for an artificial design agent?" They are also aiming to show directions for how to use EC for architecture. Together, they provide a conceptual structuring framework, which may incorporate the dynamic temporal and hierarchical relationships for reusable computational techniques targeting architectural design processes. Specific ASCs may be understood as preliminary automation frameworks, and in the long run such frameworks may be used in the development of design-specific artificial agents. As such, the ASC approach lays down a series of specific research paths, and it can be used to coordinate a unified research agenda for AD.

Currently, d_p.layout's task definition flattens several layers of the example ASC schemes into one layer. Functional zoning, circulation, and arrangement of DUs are handled together. This is not exactly congruent with the layer based decomposition scheme, and it forces the layout problem into an unnecessarily complex evolution. However, a separation of layers requires a dynamic integration mechanism, which is what was aimed in this flattening. Flattening enables a dynamic integration of these three aspects with the cost of a more complicated evolution. However, it is obvious that other dynamic integration mechanisms have to be invented, in order to keep the tasks separate, hence simple enough for evolution. With such a mechanism, site placement, massing, and facade design layers could also be added to the process, which would help in developing more convincing draft buildings. This is also necessary for a fuller demonstration of the applicability of the ASC approach.

An interesting possibility appears as developing layers as separate modules, in order to build a parallel and hierarchical system architecture, as Rodney Brooks (1999) has done with the "Subsumption architecture". This would be a natural progression for the layer based approach, which is first and foremost influenced by Brooks' horizontal decomposition scheme. These modules could be arranged as agents operating over the parts of the same model representation, as in BIM collaboration platforms. Each such module would immediately respond to the changes that have been done by other modules in its region on the common model, without needing other means of communication or the intervention of a central controller. For

simplifying the process, the interactions of the modules could be controlled within a hierarchical network as in the subsumption architecture (Brooks, 1999).

If it were possible to fulfill all the layers with EC-based approaches, an EC-based architectural design agent would be possible. This is interesting as a speculative hypothesis, although not as convincing. For such a modular agent architecture, a hybrid approach that brings EC and other approaches together appears practically more viable.

An ASC-based agent could be accepted in an office's workflow by producing a series of different tentative proposal buildings within different stylistic characteristics. However, given its technical limitations, issues like ease of use and pleasure would still be decisive in its practical success.

5.5 Theoretical Outputs and Additional Remarks

Besides practical outputs, the thesis put forward a set of illustrations that summarize its readings on related literature, which may benefit other researchers studying similar subjects. Chapter 2 was dedicated to a detailed theoretical outline concerning the relationship of design processes and Computational Design, which at the same time reveals, supports, and discusses the aims and directions adopted in the thesis. An underlying motivation for this theoretical investigation was to find out why previous approaches have had problems in satisfying the quest for artificial design intelligence.

During this exposition, we have appealed to an imaginary story, which was methodologically peculiar. The story was not related to an ethnographical or another empirical study. Yet, we have still ventured to use such an approach, because the issues that were stressed in the story were mainly supported by reliable design research literature. In other words, the fiction was rather a summary-diagram or mapping of these existing claims, but in concrete elements. Stories may be powerful descriptive tools for highly contextual fields like architectural design. They may be helpful in the presentation of a series of interlinked aspects in a strongly integrated form. In this sense, stories may be more powerful than diagrams or models, which tend to stay on a more abstract level. The important point here is to clearly separate

speculation from existing theory, for which reason the story is given only after a comprehensive introduction to the theory in overall and abstract terms.

Laboratory protocol studies, more often than not, focus on acquiring a systematic account of the molecular levels of design processes, mostly within highly constrained, short-term design situations. They should be complemented with ethnographical studies that may take into account contextual background and long-term transformations, for which story telling would be a natural outcome. Yet, the difference should be noted between a description of a real process and a diagramming through story telling; these are separate tools that may be useful in different contexts. It is not attempted to discredit graphical diagrams, mappings, models, etc. with this account. On the contrary, all these tools have their uses in theoretical examinations and the thesis have made extensive use of such tools.

A second summary was a non-reductive mapping of the basic constituents of a design situation (Figure 2.15). The mapping involved agents (designers, consultants, and other experts, users and target groups, contractors and managers, property developers and advertisers, politicians, public administrators and legislators) who access the co-evolving problem and solution spaces through viewpoints, interpretations, framings, and actions. Rather than constituting a list of atomic basic actions, each of these terms designates a separate aspect of a complex interaction.

Another summary was an abstract classification for the multifaceted evaluations required within design situations (Section 2.4.10). The main aim of this classification was to complement the technically oriented approaches to performance, which are not sufficient for architecture. The scheme classified the types of performances under four headings, i.e., prudence, well-defined performance, underdefined performance, and undefined performance.

The fourth summary generated through this investigation was a mapping of the difficulties encountered by the computational studies in design as linked to the corresponding strategies and techniques employed by human designers (Figure 2.16). The first group of problems concerns the "inherent complexity" of design situations in terms of a multiplicity of objectives, requirements, constraints, parts, and productions. Additionally, there appears an "inherent undefinedness" because of the multiplicity of viewpoints and tastes, which result in un(der)defined tasks,

incomplete information, and the contextual character of design knowledge. Design situations are essentially undefined, because until a design process starts to unfold, the required negotiations cannot take place, the tasks cannot be defined, and relevant information cannot be determined. This complex and undefined nature of design situations is the basis of our position against reductive strategies.

The second part of this mapping (Figure 2.16) attempts to identify the strategies assumed by human designers encountering these challenges. The basis of these strategies is the dynamic approaches built over a substrate of daily, embodied, situated intelligence. Interpretation and domain-specific expertise is to be built over this daily substrate and will involve, on the one hand, prefabricated, reusable strategies such as heuristics, schemata, patterns, precedents, style, etc., and on the other hand, dynamic strategies such as tentative proposals and viewpoints (re-framing), changeable motivations, goals, objectives, criteria, constraints (a practical perspectivism within an ongoing negotiation), continuous learning (eliciting new information where required), and dynamic structuring of the process through dynamic focus (decomposition + integration).

It should be noted that the empirical validity of these complementary summaries is dependent on the texts that were referred to. An additional empirical validation has not been attempted since such an endeavor would demand an altogether different orientation for the thesis. Nevertheless, with due caution, this interpretative output has been used as a substrate for answering an overall question of the thesis, i.e., how could a viable research program be determined for Artificial / Autonomous / Automated design (AD)?

The rather general component of this response is a characterization of AD as a research program, for which a set of basic tenets has been proposed: (1) current human strategies and behaviors in design are the proper response to design situations, (2) design computation takes place in a design environment where the humans and CAD tools operate together, and (3) while weak-AD is the practical paradigm for Computational Design, strong-AD may remain an ultimate target.

The only viable way towards this ultimate aim is the cooperative human-tool complexes that would evolve together, and get to know each other towards a level of mutual understanding. The basic requirements for this target are machine-learning

technologies and the voluntary acceptance of the artificial agents for cooperation. In this context, following Negroponte's ideas, the thesis assumed that the AD tools should be intuitive and insightful companions for the human designers. Thus, the overall task appears as to develop more intelligent assistants that would be able to work smoothly with the human agents while learning from them. In other words, to develop human-tool complexes that would function better or at least more pleasurably for the human counterpart. The practical proposals of the thesis have been based on this research program.

Throughout this thesis, a side theme has been the possible role of EC in the collaboration of design agents. The study examined EC's potential contributions on this quest by focusing on its limitations, potentials, and requirements for design tasks, through both theoretical examinations and applications. The question concerning the required collaboration scheme for the utilization of EC within design tasks is answered by a generic task structure (Figure 3.2). The task structure locates complex EAs one step above the simpler ones, yet posits separate intelligent agents as necessary for the guiding of the process. A multi-level learning EC is hypothesized (Figure 3.5) during the discussion of the complex EAs, which would gather all its constituents within hierarchical levels of evolution.

For human-machine relationships, the AD program offers a conceptual background, the evolutionary task structure (Figure 3.2) provides a general framework, and the d_p.layout demonstrates a practical example. However, although a collaboration scheme has been given for the d_p.graphics in Chapter 4 (Figure 4.24), it would be hard to claim that human-to-human collaboration has been adequately addressed within this thesis. Indeed, together with the d_p.layout related themes, the two d_p.layout workshops (Chapter 4) had been equally devoted to the exploration of human collaboration, yet the workshops were mostly fruitless in this respect. The only viable proposal for collaboration has remained an open source development of the target pools, which, nonetheless exhibits a potential to make architectural knowledge more accessible.

Because the research has been based on EC technologies, it comes to ask whether there may be a continuous research path from the current state of EC to intelligent design agents. EAs are soft-computing environments, which operate mostly through randomized operators and with a minimum of intelligence. With the incorporation of

a more multifaceted intelligence, the process would better be called design, instead of evolution. Evolution is not intelligent design[27]. However, EC may be used by intelligent agents, or together with intelligent technologies to carry out design. Because it can operate with minimal intelligence, EC makes it possible to use limited, partial, weak artificial intelligence technologies for design problems and its utility stems exactly from this point: while we do not have an advanced AI for design, EC caters for the continuity of research.

Suppose that there are intelligent systems capable of recognizing or evaluating partial aspects within design situations. Such systems could be used as evaluation procedures for an EA in order to assess the potentials of specific design proposals. For example, a series of artificial neural networks (ANN) could be used for recognizing separate feature distributions of architectural layouts. Recognition systems could also be used to monitor the progression of an EA, or on a higher level, there could be intelligent agents, which could monitor the functioning of an EA within a larger design situation, in other words which could recognize the state of an EA within a design context. At each level, these evaluations would provide us with rationales to guide the process. These systems could focus on the recognition of states, in terms of their similarity to a set of prototypes. Consider a scenario where the capability of such state recognition systems is gradually being improved. On the intelligent end of this gradual process, there are hypothetical systems that are capable of recognizing and evaluating states of EC systems within design situations.

While the intelligence of the system is relatively low, as is the case with the d_p.layout system, current EC approaches would be utilized, and the products would be developed through mostly unintelligent stochastic operations. As the embedded or guiding intelligence improves, micro and macro operations may begin to be based on a series of rationales, which would help reduce the number of the alternative individuals at each generation. The process gradually starts to resemble an interactive EA, ultimately arriving at a state where intelligent agents carry out the process through a minimal number of alternative proposals, yet with evaluations that are more comprehensive and sensitive. At this point, the process would quit being an EA

[27] For an alternative discussion of this distinction, see Hanna (2010).

and would become design proper. Due to the basic characteristics of EC, this scenario may be commenced with minimal intelligence. And if a gradual improvement in intelligent systems is possible, we can claim according to this scenario that there should be a continuity towards intelligent design; with gradual increases at all aspects of the process; i.e., intelligent operative moves, better evaluation, and dynamic process structuring.

It should be noted while concluding that, at some points, this research has become too broad or vague, and somewhat troublesome for the researcher; as the character of the tasks made it difficult to confine the research path and brought forward a continuous need to develop and adapt specific methods and approaches at each new step. The nature of the problem demanded a high amount of intellectual effort. Furthermore, because of the continuous postponement of practical outputs, this has not been a comfortable research. However, at the same time, these basic problems have constituted the greater part of the excitement for the study, which provided a feeling of persistence, rather than the contrary. Although tiresome, continuously renewed problems also mean renewed challenges, and the exploratory attitude stimulates continuous learning and intellectual development.

REFERENCES

Achten, H. H. (2009). Experimental design methods, a review. *International Journal of Architectural Computing, issue 04, volume 07*.

Akin, Ö. (2001). Simon Says: Design is Representation. (Draft version). Retrieved from http://www.andrew.cmu.edu/user/oa04/Papers/AradSimon.pdf Date retrieved: November 2011.

Akin, Ö., Dave, B., and Pithavadian, S. (1992). Heuristic generation of layouts (HeGeL): based on a paradigm for problem structuring. *Environment and Planning B: Planning and Design 19,* 33 – 59. doi:10.1068/b190033.

Alexander, C. (1966). *Notes on the synthesis of form*. Cambridge: Harvard University Press.

Alexander, C. (1977). *A Pattern Language: Towns, Buildings, Construction*. Oxford University Press.

Alexander, C. (1984). The State of the Art in Design Methods. In N. Cross (Ed.), *Developments in Design Methodology*. Wiley, Chichester, pp. 309-316.

Alexiou, K., Zamenopoulos, T., and Johnson, J. H. (2009). Exploring the neurological basis of design cognition using brain imaging: some preliminary results. *Design Studies, 30* (2009) 623-647.

Ang, M. C., Chau, H. H., Mckay, A., and De Pennington, A. (2006). Combining evolutionary algorithms and shape grammars to generate branded product design. In J.S. Gero (Ed.), *Design Computing and Cognition '06*, 521–539.

Back, T., Fogel, D.B., and Michalewicz, Z. (Eds.) (2000). *Evolutionary Computation 2: Advanced Algorithms and Operators.* IOP Publishing Ltd, Bristol and Philadelphia.

Banerjee, A., Quiroz, J. C., and Louis, S. J. (2008). A model of creative design using collaborative interactive genetic algorithms. In J. S. Gero, and A. K. Goel (Eds.), *Design Computing and Cognition '08*, pp. 397-416 Springer Science + Business Media B.V.

Bayazıt, N. (2004). Investigating design: a review of forty years of design research. *Design Issues, Volume 20, Number 1* Winter 2004.

Baykan, C. A. and Fox, M. S. (1997). Spatial synthesis by disjunctive constraint satisfaction. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing, Volume 11 / Issue 04 /* September 1997, pp 245-262.

Bentley, P. J. (Ed.) (1999). *Evolutionary Design by Computers*. Morgan Kaufman Publishers, San Francisco CA.

Bentley, P. J. (2001). Ten Steps to Make a Perfect Creative Evolutionary Design System. *Proceedings of GECCO 2001*.

Bentley, P. J. and Corne, D. W. (Eds.) (2002). *Creative Evolutionary Systems*. Academic Press, London, UK

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning.* Springer Science+Business Media, LLC.

Bittermann, M. (2010). *Intelligent Design Objects: A Cognitive Approach for Performance-Based Design*. Phd Dissertation, TUDelft.

Boden, M. A. (1998). Creativity and artificial intelligence. *Artificial Intelligence*, *103* (1998) 347-356.

Boden, M. A. (2004). *The Creative Mind Myths and Mechanisms*. Routledge, (second edition).

Broadbent, G. and Ward, A. (1969). *Design Methods in Architecture*. Lund Humphries.

Broadbent, G. (1973). *Design in Architecture: Architecture and The Human Sciences*. John Wiley & Sons, London.

Brooks, R.A. (1999). *Cambrian intelligence: the early history of the new AI*. MIT Press, Cambridge, Mass.

Buchanan, R. (1992). Wicked problems in design thinking. *Design Issues*, *Vol. 8,* No. 2 (Spring, 1992), pp. 5-21.

Caldas, L. G. (2003). Shape generation using pareto genetic algorithms. *CAADRIA 2003*.

Caldas, L. G. (2005). Three-dimensional shape generation of low-energy architecture solutions using Pareto GA's. *Proceedings of ECAADE'05*, Lisbon, September 21-24, 2005, pp. 647-654.

Caldas, L. G. (2006). GENE_ARCH: An evolution-based generative design system for sustainable architecture. I. F. C. Smith (Ed.), *EG-ICE 2006, LNAI 4200*, pp. 109 – 118, 2006.

Caldas, L. G. (2008). Generation of energy-efficient architecture solutions applying GENE_ARCH: An evolution-based generative design system. *Advanced Engineering Informatics, Volume 22,* Issue 1 (January 2008).

Caldas, L. G. and Norford, L. K. (2002). A design optimization tool based on a genetic algorithm. *Automation in Construction*, *11* (2002) 173–184.

Caldas, L. G. and Norford, L. K. (2003). Genetic Algorithms for Optimization of Building Envelopes and the Design and Control of HVAC systems. *Journal of Solar Energy Engineering, August 2003, Vol. 125*.

Caldas, L. G. and Rocha, J. (2008). A generative design system applied to Siza's school of architecture at Oporto. In J. S. Gero, S. Chase and M. Rosenman (Eds), *CAADRIA2001*, Key Centre of Design Computing and Cognition, University of Sydney, 2001, pp. 253-264.

Ceccato, C., (1998). MICROGENESIS: The architect as toolmaker: computer-based Generative design tools and methods. *Generative Art*, Milan, Italy, December, 1998.

Chandrasekaran, B. (1990). Design problem solving: a task analysis. *AI Magazine, Winter, 1990.*

Chouchoulas, O. (2003). *Design Shape Evolution: An Algorithmic Method for Conceptual Architectural Design Combining Shape Grammars and Genetic Algorithms*. Phd dissertation, Centre for Advanced Studies in Architecture Department of Architecture and Civil Engineering University of Bath.

Chouchoulas, O. and Day, A. (2007). Design exploration using a shape grammar with a genetic algorithm. *Open House International, Vol 32,* No.2, June 2007.

Clark, A. and Chalmers, D. J. (1998). The Extended Mind. Reprinted in Grim, P. (Ed.), *The Philosopher's Annual, vol XXI*.

Cross, N. (1977). *The Automated Architect*. Pion, London.

Cross, N. (Ed.) (1984). *Developments in Design Methodology*. Wiley, Chichester.

Cross, N. (1997). Descriptive models of creative design: application to an example. *Design Studies, 18* (1997) 427—455

Cross, N. (2000). *Engineering Design Methods: Strategies for Product Design*. 3rd ed. Wiley.

Cross, N. (2004). Expertise in design: an overview. *Design Studies, 25* (2004) 427–441.

Cross, N. (2006). *Designerly Ways of Knowing*. Springer,London.

Cross, N. (2011). *Design Thinking: Understanding How Designers Think and Work*. Bloomsbury Academic, Oxford; New York.

Cross, N. and Cross, A. C. (1995). Observations of teamwork and social processes in design. *Design Studies, 16* (1995).

Cross, N., Christiaans, H., Dorst, K. (Eds.) (1996). *Analysing Design Activity*. Wiley, Chichester.

Da Silva Garza, A.G. and Lores, A. Z. (2008). An evolutionary process model for design style imitation. In J.S. Gero and A.K. Goel (Eds.), *Design Computing and Cognition '08*, Springer Science + Business Media B.V. 2008.

Damski, J. C. and Gero, J. S. (1997). An evolutionary approach to generating constraint-based space layout topologies. In R. Junge (Ed.), *CAADFutures 1997*, Kluwer, Dordrecht. pp. 855-864.

Darke, J. (1979). The Primary Generator and the Design Process. *Design Studies, Vol. 1,* No. 1, July 1979.

Datta, R., Joshi, D., Li, J., and Wang, J. Z. (2008). Image retrieval: ideas, influences, and trends of the new age. *ACM Computing Surveys, Vol. 40,* No. 2, Article 5, Publication date: April 2008.

Dawkins, R. (1996). *The Blind Watchmaker*. Penguin Books, pp. 43-74.

De Jong, K. A. (2006). *Evolutionary Computation: A Unified Approach*. The MIT Press.

De Jong, T. M. and Van Der Voordt, D. J. M. (Eds.) (2002). *Ways to Study and Research: Urban, architectural and technical design*. DUP Science.

De Landa, M. (2004). Deleuze and the use of genetic algorithms in architecture. in Leach, N., Turnbull, D. (Eds.), *Digital tectonics*. Wiley-Academy, Chichester.

Dennett, D. C. (1991). *Consciousness Explained*. Back Bay Books, Little, Brown and Company, New York, Boston, London.

Dillenburger, B., Braach, M., and Hovestadt, L. (2009). Building design as an individual compromise between qualities and costs a general approach for automated building generation under permanent cost and quality control. In T. Tidafi, and T. Dorta (Eds.), *Joining Languages, Cultures and Visions: CAADFutures 2009*.

Dorst, K. (2004). On the Problem of Design Problems: problem solving and design expertise. *J. of Design Research*, Vol. 4, No.2.

Dorst, K. (2006). *Understanding Design*. Gingko Press (2nd edition), Corte Madera, CA.

Dorst, K. (2006b). Design Problems and Design Paradoxes. *Design Issues, Volume 22*, Number 3 Summer.

Dorst, K. and Cross, N. (2001). Creativity in the design process: co-evolution of problem-solution. *Design Studies, Vol. 22*, No. 5, pp. 425-437.

Dorst, K. and Dijkhuis, J. (1995). Comparing paradigms for describing design activity. *Design Studies, 16* (1995) 261-274.

Dreyfus, H. L. (1972). *What Computers Can't Do: A Critique of Artificial Reason*. Harper & Row, First edition.

Dreyfus, H. L. (1999). *What Computers Still Can't Do: A Critique of Artificial Reason*. The MIT Press, Cambridge, Massachusetts, London, England, sixth edition.

Dreyfus, H. L., Dreyfus, S. E., and Athanasiou, T. (1986). *Mind over machine: the power of human intuition and expertise in the era of the computer*. Free Press, New York.

Duarte, J. P. (2001). *Customizing Mass Housing: A Discursive Grammar for Siza's Malagueira Houses*. Phd dissertation. Massachusetts Institute of Technology. Dept. of Architecture.

Eastman, C. M. (1973). Automated space planning. *Artificial Intelligence, 4* (1973), 41—64.

Eastman, C., Teicholz, P., Sacks, R., and Liston, K. (2011). *BIM Handbook, A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers, and Contractors*. John Wiley & Sons, Inc. (second edition).

Eiben, A. E., Nabuurs, R., and Booij, I. (2002). The Escher Evolver: Evolution to the People. In P. J. Bentley, and D. W. Corne (Eds.), *Creative Evolutionary Systems*. Academic Press, London, UK.

Eiben, A. E. and Smit, S. K. (2012). Evolutionary Algorithm Parameters and Methods to Tune Them. in Hamadi, Y., Monfroy, E., and Saubion, F. (Eds.), *Autonomous Search*, Springer, Berlin, Heidelberg, Chapter 2.

Eiben, A. E. and Smith, J. E. (2003). *Introduction to Evolutionary Computing*. Springer, New York.

Eliasmith, C., Stewart, T. C., Choo, X., Bekolay, T., DeWolf, T., Tang, Y., and Rasmussen, D. (2012). A Large-Scale Model of the Functioning Brain, *Science 338* (6111) (November 30), 1202–1205, doi:10.1126/science.1225266.

Fernando, R., Drogemuller, R., Salim, F., and Burry, J. (2010). Patterns, heuristics for architectural design support making use of evolutionary modelling in design; In B. Dave, A. I. Li, N. Gu, H. J. Park (Eds.), *New Frontiers: Proceedings of the 15th International Conference on Computer-Aided Architectural Design Research in Asia CAADRIA 2010*, 283–292.

Flack, W. J. and Ross, B. J. (2013). Evolution of Architectural Floor Plans. In Esparcia-Alcázar, A. I. (Ed.), *Applications of Evolutionary Computation, 16th European Conference, EvoApplications 2013*, Vienna, Austria, April 3-5, 2013.

Flemming, U. and Woodbury, R. (1995). Software Environment to Support Early Phases in Building Design (SEED): Overview. *Journal of Architectural Engineering*, *1* (4), 147–152.

Frazer J. H. (1995). *An Evolutionary Architecture*. Architectural Association, London.

Frazer J. H., Frazer J., Liu X., Tang, M., Janssen, P. (2002). Generative and evolutionary techniques for building envelope design. *Generative art 2002*.

Gedenryd, H. (1998). *How Designers Work*. Lund University.

Geigel, J. and Loui, A. (n.d.). Automatic Page Layout Using Genetic Algorithms for Electronic Albuming. Research and Development, Eastman Kodak Company, Rochester, NY 14650-1816.

Gero, J. S. (1990). Design prototypes: a knowledge representation schema for design. *AI Magazine*, *Volume 11* Number 4 (1990).

Gero, J. S. (1994). Towards a model of exploration in computer-aided design. *Formal Design Methods for CAD*, Gero, J. S. and Tyugu, N. (Eds.), North-Holland, Amsterdam, pp. 315–336.

Gero, J. S. (1998). Conceptual designing as a sequence of situated acts. In I. Smith (Ed.), *Artificial Intelligence in Structural Engineering*. Berlin, Springer- Verlag: 165–177.

Gero, J. S. (1999). Representation and reasoning about shapes: cognitive and computational studies in visual reasoning in design. In C. Freska, D.

M. Mark (Eds.), *Spatial Information Theory - Cognitive and Computational Foundations of Geographic Information Science*. *International Conference COSIT '99*, Stade, Germany, August 25-29, 1999.

Gero, J. S. (2002). Computational Models of Creative Designing Based on Situated Cognition. *Proceedings of the 4th conference on Creativity & cognition*. C&C '02.

Gero, J. S. and Damski, J. (1999). Feature-based qualitative modeling of objects. In G. Augenbroe, and C. Eastman (Eds.), *Computers in Building*, Kluwer, Boston, pp. 309-320.

Gero, J. S. and Kannengiesser, U. (n.d.). An Ontological Account of Donald Schön's Reflection in Designing. *International Journal of Design Sciences and Technology*, *Volume: 15* Number: 2 Pages: 77-90.

Gero, J. S. and Kannengiesser, U. (2004). Modelling Expertise of Temporary Design Teams. *Journal of Design Research*, *2004 - Vol. 4*, No.2.

Gero, J. S. and Kannengiesser, U. (2007). A function–behavior–structure ontology of processes. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing, (2007), 21*, 379–391.

Gero, J. S. and Kazakov, V. A. (1998). Evolving design genes in space ayout planning problems. *Artificial Intelligence in Engineering*, *12* (1998) 163-176.

Gero, J. S. and Kazakov, V. (2001). Entropic-based similarity and complexity measures of 2D architectural drawings. In J. S. Gero, B. Tversky, and T. Purcell (Eds.), *Visual and Spatial Reasoning in Design II*, Key Centre of Design Computing and Cognition, University of Sydney, Australia, pp. 147-161.

Gero, J. S. Louis S., and Kundu, S. (1994). Evolutionary learning of novel grammars for design improvement. *AI EDAM*, *8* (2):83-94.

Gero, J. S. and Park, S. H. (1997). Computable feature-based qualitative modeling of shape and space. In R. Junge (Ed.), *CAADFutures 1997*, Kluwer, Dordrecht. pp. 821-830.

Gero, J. S. and Saunders, R. (2000). Constructed representations and their functions in computational models of designing. *Proceedings of CAADRIA 2000.*

Gero, J. S. and Saunders, R. (2002). Curious agents and situated design evaluations. In J. S. Gero and F. M. T. Brazier (Eds.) *Agents in Design*. Key Centre of Design Computing and Cognition, University of Sydney, pp. 133-149.

Gero, J. S. and Schnier, T. (1995). Evolving representations of design cases and their use in creative design. *Third International Conference on Computational Models of Creative Design*.

Gero, J. S. and Smith, G. J. (2009). Context, situations, and design agents. *Knowledge-Based Systems*, *22* (2009) 600–609.

Gero, J. S. and Tyugu, E. (Eds.) (1994). Formal design methods for CAD. *Proceedings of the IFIP TC5/WG5.2 Workshop on Formal Design Methods for CAD*. Tallinn, Estonia, 16-19 June 1994.

Goldschmidt, G. (2006). Quo Vadis, Design Space Explorer? *AIE EDAM: Artificial Intelligence for Engineering Design, Analysis, and Manufacturing*. *20*, no. 02 (2006): pp.105–111.

Gu, Z., Tang M. X., and Frazer, J. H. (2006). Capturing aesthetic intention during interactive evolution. *Computer-Aided Design*, *38* (2006) 224–237.

Hanna, S. (2005). Automated representation of style by feature space archetypes: distinguishing spatial styles from generative rules. *International Journal of Architectural Computing*, *issue 01, volume 05*.

Hanna, S. (2006). Representing style by feature space archetypes, description and emulation of spatial styles in an architectural context. In J.S. Gero (Ed.), *Design Computing and Cognition'06*, 2006, 3–22.

Hanna, S. (2007a). Defining implicit objective functions for design problems. *GECCO'07*, July 7–11.

Hanna, S. (2007b). Representation and generation of plans using graph spectra. In: Kubat, A.S., Ertekin, O., Guney, Y.I., and Eyuboglu, E., (eds.) *6th International Space Syntax Symposium*, 12-15 Jun 2007, Istanbul, Turkey.

Hanna, S. (2010). Intelligent Design: Going back to Darwin for a Better Computational Model of Creation. *Proceedings of 3rd Design Creativity Workshop*, July 11, 2010, Stuttgart, Germany.

Hanna, S. (2012). The Hard Problem of Design. *AI EDAM-Artificial Intelligence for Engineering Design Analysis and Manufacturing*, *26* (1) 15 - 15.

Hart, D. A. (2007). Toward Greater Artistic Control for Interactive Evolution of Images and Animation. In M. Giacobini et al. (Eds.), *EvoWorkshops 2007, LNCS 4448*, pp. 527–536, 2007. Springer-Verlag.

Hewgill, A. and Ross, B. J. (2004). Procedural 3D texture synthesis using genetic programming. *Computers & Graphics*, *28* (2004) 569–584.

Inoue, M. and Takagi, H. (2009). EMO-based architectural room floor planning. *Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on. IEEE*, pp. 518–523.

Horst, S. (2011). The Computational Theory of Mind. *The Stanford Encyclopedia of Philosophy*. E. N. Zalta (ed.), Spring 2011 Edition, Retrieved from http://plato.stanford.edu/archives/spr2011/entries/computational-mind/ Date retrieved, June, 2013.

Janssen, P., Frazer, J., and Tang, M. (2002). Evolutionary design systems and generative processes. *Applied Intelligence, 16*, 119–128, 2002.

Janssen, P. H. T. (2004). A design method and computational architecture for generating and evolving building designs. Phd Dissertation, Hong Kong Polytechnic University.

Janssen, P. H. T. (2006). A generative evolutionary design method. *Digital Creativity*, *17*:1, 49 – 63.

Janssen, P. H. T. (2006b). The role of preconceptions in design. J.S. Gero (Ed.), *Design Computing and Cognition '06*, 365–383.

Jo, J. H. and Gero, J. S. (1998). Space layout planning using an evolutionary approach. *Artificial Intelligence in Engineering, 12* (1998) 149-162.

Jones, J. C. (1984). How my thoughts about second-generation design methods have changed over the years. In N. Cross (Ed.), *Developments in Design Methodology*. Wiley, Chichester, pp. 329-336.

Jones, J. C. (1992). *Design Methods*. Second edition, Van Nostrand Reinhold, New York.

Jupp, J. R. and Gero, J. S. (2006). A qualitative feature-based characterization of 2D architectural style. *Journal of The American Society for Information Science and Technology, 57* (11):1537–1550, 2006.

Kalay, Y. E. (Ed.) (1992). *Evaluating and Predicting Design Performance*. Wiley, New York.

Kalay, Y. E. (2004). *Architecture's New Media: Principles, Theories, and Methods of Computer-Aided Design*. MIT Press.

Keane, A. J. and Brown, S. M. (1996). The design of a satellite boom with enhanced vibration performance using genetic algorithm techniques. In I.C. Parniee, Ed., *Proceedings of the Conference on Adaptive Computing in Engineering Design and Control 96. P.E.D.C*: Plymouth, 1996. pp. 107-113.

Kolarevic, B. (Ed.) (2003). *Architecture in The Digital Age: Design and Manufacturing.* Spon Press, New York.

Lakoff, G. and Johnson, M. (1999). *Philosophy in the flesh: the embodied mind and its challenge to Western thought*. Basic Books, New York.

Lawson, B. (2004). *What Designers Know*. Elsevier / Architectural Press, Amsterdam.

Lawson, B. (2004b). Schemata, gambits and precedent: some factors in design expertise. *Design Studies, 25* (2004) 443–457.

Lawson, B. (2005). *How Designers Think: The Design Process Demystified*. Architectural Press (fourth edition).

Lawson, B. and Dorst, K. (2009). *Design Expertise*. Architectural Press.

Lewis, M. (2008). Evolutionary visual art and design. In Romero, J., and Machado, P. (Eds.), *The Art of Artificial Evolution: A Handbook on Evolutionary Art and Music,* Springer-Verlag.

Liggett, R. S. (2000). Automated facilities layout: past, present and future. *Automation in Construction, 9* 2000. 197–215.

Littlefield, D. (Ed.) (2008). *Space Craft: Developments in Architectural Computing*. RIBA Pub., London.

Liu, H. and Tang, M. (2006). Evolutionary design in a multi-agent design environment. *Applied Soft Computing, 6* (2006) 207–220.

Liu, H., Tang, M., and Frazer J. H. (2002). Supporting evolution in a multi-agent cooperative design environment. *Advances in Engineering Software, 33* (2992) 319-328.

Lok, S. and Feiner, S. (2001). A survey of automated layout techniques for ınformation presentations. *2001 SmartGraphics '01,* Hawthorne, NY USA.

Lok, S., Feiner, S., and Ngai, G. (2001). Evaluation of visual balance for automated layout. *IUI'04*, January 13–16, 2004, Madeira, Funchal, Portugal.

Luger, F. (2004). *Artificial Intelligence Structures and Strategies for Complex Problem Solving*. Addison Wesley (fifth edition).

Machado, P., Romero, J., Cardoso, A., and Santos, A. (2005). Partially ınteractive evolutionary artists. *New Generation Computing, 23* (2005)143-155.

Machado, P., Romero, J., Santos, M. A., Cardoso, A., and Manaris, B. (2004). Adaptive critics for evolutionary artists. G.R. Raidl et al. (Eds.), *EvoWorkshops 2004, LNCS 3005*, pp. 437–446, 2004.

Maher, M. L. (2000). A Model of Co-evolutionary Design. *Engineering with Computers, (2000) 16*: 195–208.

Maher, M. L. (2007). Blurring the boundaries. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing, 2007, 21*, 7–10.

Maher, M. L., Poon, J., and Boulanger, S. (1996). Formalising design exploration as co-evolution: a combined gene approach. In J. S. Gero, and F. Sudweeks (Eds.) *Advances in Formal Design Methods for CAD*, Chapman and Hall, London, UK.

Masui, T. (1994). Evolutionary learning of graph layout constraints from examples. *UIST '94 Annual Symposium on User Interface Software and Technology*, November 2-4, 1994.

McCormack, J. (2008). Facing the Future: Evolutionary possibilities for human-machine creativity. In J. Romero; P. Machado (Eds.), *The Art of Artificial Evolution: A Handbook on Evolutionary Art and Music*; Springer-Verlag.

McDonnell, J. and Lloyd, P. (Eds.) (2009). *About: Designing: analysing design meetings*. CRC Press, Boca Raton.

Michalek, J., Choudhary, R., and Papalambros, P. (2002). Architectural layout design optimization. *Engineering Optimization 34*, 461–484. doi:10.1080/03052150214016.

Michalewicz, Z. and Fogel, D. B. (2004). *How to Solve It: Modern Heuristics*, Second edition, Springer, Berlin; New York.

Negroponte, N. (1970). *The Architecture Machine: Toward a More Human Environment*. MIT Press.

Negroponte, N. (ed.) (1975). *Reflections on Computer Aids to Design and Architecture*. Petrocelli/Charter, New York.

Negroponte, N. (2011). Towards a Humanism Through Machines. in Menges, A., and Ahlquist, S. (Eds.), *Computational Design Thinking: Computation Design Thinking*. 1st ed. Wiley.

Newell, A. and Simon, H. A. (1972). *Human Problem Solving*. Prentice-Hall.

Niedderer, K. (2007). Mapping the Meaning of Knowledge in Design Research. *Design Research Quarterly, 2* (2), April.

Nilsson, N. J. (1998). *Artificial Intelligence: A New Synthesis*. Morgan and Kauffman.

Nilsson, N. J. (2009). *The Quest for Artificial Intelligence*. Cambridge University Press.

Nishino, H., Sueyoshi, T., Kagawa, T., and Utsumiya, K. (2008). An interactive 3D graphics modeler based on simulated human immune system. *Journal of Multimedia, vol. 3*, no. 3, july 2008.

Oxman, R. and Gero, J. S. (1987). Using an expert system for design diagnosis and design synthesis. *Expert Systems 4*, 4–14.

Park, S-H. and Gero, J. S. (2000). Categorisation of shapes using shape features. In Gero, J. S. (Ed.), *Artificial Intelligence in Design'00*, Kluwer, Dordrecht, pp.203-223.

Pfeifer, R. and Bongard, J. (2006). *How the Body Shapes the Way We Think: A New View of Intelligence*. A Bradford Book, Cambridge, Mass.

Pollack, S. (Dir.) (2006). *Sketches of Frank Gehry* (Documentary film). American Masters, LM Media, Mirage Enterprises, Public Broadcasting Service (PBS), SP Architecture, WNET Thirteen.

Psarra, S. and Grajewski T. (2001). Describing shape and shape complexity using local properties. *Proceedings of the 3rd International Space Syntax Symposium*, Atlanta 2001.

Rittel, H. W. J. and Webber, M. M. (1973). Dilemmas in a General Theory of Planning. *Policy Sciences, 4* (1973), 155-169.

Rivard, H. and Fenves, S. J. (2000). SEED-Config: A case-based reasoning system for conceptual building design. *AI EDAM, 14*, 415–430.

Rodrigues, E., Gaspar, A.R., and Gomes, Á. (2013a). An approach to the multi-level space allocation problem in architecture using a hybrid evolutionary technique. *Automation in Construction 35*, 482–498. doi:10.1016/j.autcon.2013.06.005.

Rodrigues, E., Gaspar, A.R., and Gomes, Á. (2013b). An evolutionary strategy enhanced with a local search technique for the space allocation problem in architecture, Part 1: Methodology. *Computer-Aided Design 45*, 887–897. doi:10.1016/j.cad.2013.01.001.

Rodrigues, E., Gaspar, A.R., and Gomes, Á. (2013c). An evolutionary strategy enhanced with a local search technique for the space allocation problem in architecture, Part 2: Validation and performance tests. *Computer-Aided Design 45*, 898–910. doi:10.1016/j.cad.2013.01.003.

Romero, J. and Machado, P. (Eds.) (2008). *The Art of Artificial Evolution: A Handbook on Evolutionary Art and Music*; Springer-Verlag.

Rosenman, M. A. (1996). The Generation of Form Using an Evolutionary Approach. In Gero, J. S. and Sudweeks, F. (Eds.), *Artificial Intelligence in Design '96*. Springer Netherlands, pp. 643–662.

Rosenman, M. A. (2000). Case-based evolutionary design. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing, 2000, 14*, 17–29.

Rosenman, M. A. and Saunders, R. (2003). Self-regulatory hierarchical coevolution. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing, 2003, 17*, 273–285.

Ross, B.J., Ralph, W., and Zong, H. (2006). Evolutionary image synthesis using a model of aesthetics. In G.G. Yen, S.M. Lucas, G. Fogel, G. Kendall, R., Salomon, B.T. Zhang, C.A.C. Coello, and T.P., Runarsson, (eds.), *Proceedings of The 2006 IEEE Congress on Evolutionary Computation*, (Vancouver, BC, Canada). IEEE Press, 2006, 1087–1094.

Rowe, P. G. (1987). *Design Thinking*. MIT Press, Cambridge.

Russell, S. J. and Norvig, P. (2010). *Artificial Intelligence: A Modern Approach*. Prentice Hall (third edition).

Schön, D. A. (1983). *The Reflective Practitioner.* Basic Books, New York.

Schön, D. A. (1984). The architectural studio as an exemplar of education for reflection-in-action. *Journal of Architectural Education, Vol. 38,* No. 1 (Autumn, 1984), pp. 2-9.

Schön, D. A. and Wiggins, G. (1992). Kinds of seeing and their functions in designing. *Design Studies, Vol 13* No 2 April 1992.

Simon, H. A. (1955). A Behavioral Model of Rational Choice. *The Quarterly Journal of Economics, Vol. 69*, No. 1. (Feb., 1955), pp. 99-118.

Simon, H. A. (1970). Style in Design. *Proceedings of the 2nd Annual Environmental Design Research Association Conference*, EDRA 2.

Simon, H. A. (1973). The Structure of Ill Structured Problems. *Artificial Intelligence, 4* (3): 181–201.

Simon, H. A. (1975). Style in design. *Spatial synthesis in computer-aided building design, 9*, 287–309.

Simon, H. A. (1992). Rational Decision-making in Business Organization. Economic Sciences, 1969-1980. in *The Sveriges Riksbank (Bank of Sweden) Prize in Economic Sciences in Memory of Alfred Nobel 1*, pp. 343-371 (Transcript of Nobel Memorial Lecture on 8 December, 1978).

Simon, H. A. (1996). *The Sciences of The Artificial*. MIT Press (third edition), Cambridge, Mass.

Sims, K. (1991). Artificial evolution for computer graphics. *ACM Computer Graphics, 25* (4): 319–328.

Smeulders, A. W. M., Worring, M., Santini, S., Gupta, A., and Jain, R. (2000). Content-based image retrieval at the end of the early years. *IEEE Transactions On Pattern Analysis And Machine Intelligence, VOL. 22*, NO. 12, December 2000.

Smith, J. R. and Chang, S.-F. (1999). Querying by color regions using the visualseek content-based visual query system. *Multimedia Systems, 7*: 129–140 (1999).

Soddu, C. (2002). Recognizability of the idea: the evolutionary process of argenia. In P. J. Bentley, and D.W. Corne (Eds.) *Creative Evolutionary Systems*. Academic Press, London, UK, pp. 109-128.

Spiller, N. (2008). *Digital Architecture Now: A Global Survey of Emerging Talent*. Thames & Hudson, London.

Stiny, G. (2006). *Shape: Talking About Seeing and Doing*. MIT, Cambridge, Mass.; London.

Todd, S. and Latham, W. (1992). *Evolutionary Art and Computers*. Academic Press.

Turrin, M., von Buelow, P., Kilian, A., and Stouffs, R. (2011). Performative skins for passive climatic comfort: A parametric design process. *Automation in Construction*, doi:10.1016/j.autcon.2011.08.001

Turrin, M., von Buelow, P., and Stouffs, R. (2011). Design explorations of performance driven geometry in architectural design using parametric modeling and genetic algorithms. *Advanced Engineering Informatics*, doi:10.1016/j.aei.2011.07.009

Url-1 <*http://www.usgbc.org/leed*>, date retrieved 01.02.2015.

Url-2 <*http://www.breeam.org/*>, date retrieved 01.02.2015.

Url-3 <*http://www.chaosgroup.com/en/2/index.html*>, date retrieved 01.02.2015.

Url-4 <*http://www.luxrender.net*>, date retrieved 01.02.2015.

Url-5 <*http://www.autodesk.com/products/autodesk-autocad/overview*>, date retrieved 01.02.2015.

Url-6 <*http://sitemaker.umich.edu/soar/home*>, date retrieved, April 2013.

Url-7 <*http://act-r.psy.cmu.edu/*>, date retrieved, May 2013.

Url-8 <*http://brain-i-nets.kip.uni-heidelberg.de/*>, date retrieved, May 2013.

Url-9 <*http://bluebrain.epfl.ch/*>, date retrieved, May 2013.

Url-10 <*http://www.stanford.edu/group/brainsinsilicon*>, date retrieved, May 2013.

Url-11 <*http://www.humanconnectomeproject.org*>, date retrieved, May 2013.

Url-12 <*http://www.cyc.com/platform/opencyc*>, date retrieved, May 2013.

Url-13 <*http://dbpedia.org/About*>, date retrieved, May 2013.

Url-14 <*http://wordnet.princeton.edu/*>, date retrieved, May 2013.

Url-15 <*http://www.kurzweilcyberart.com*>, date retrieved 01.02.2015.

Url-16 <*http://archive.org/details/sims_evolved_virtual_creatures_1994*>, date retrieved, May 2013.

Url-17 <*http://www.phy.syr.edu/courses/mirror/biomorph/*>, date retrieved, May 2013.

Url-18 <*http://libspatialindex.github.io/*>, date retrieved, May 2013.

Veltkamp, R. C. and Tanase., M. (2002). *Content-based image retrieval systems: A survey*. Technical Report TR UU-CS-2000-34 (revised version), Department of Computing Science, Utrecht University, October 2002.

Von Buelow, P. (2007). Advantages of evolutionary computation used for exploration in the creative design process. *Journal of Integrated Systems, Design, and Process Science, September 2007, Vol. 11*, No. 3, pp. 18.

Wang, W., Rivard, H., and Zmeureanu, R. (2006). Floor shape optimization for green building design. *Advanced Engineering Informatics*, *20* (2006) 363–378.

Wiens, A. L. and Ross, B.J. (2002). Gentropy: evolutionary 2D texture generation. *Computers and Graphics*, *26* (1): 75–88, 2002.

Wilson, M. (2002). Six views of embodied cognition. *Psychonomic Bulletin & Review*, *9*, 625–636.

Wittgenstein, L. (1999). *Philosophical Investigations*. Translated by G. E. M. Anscombe, Blackwell Publishers.

Wittgenstein, L. (2003). *Tractatus Logico-Philosophicus*. Routledge, London; New York.

Wolfram, S. (2002). *A New Kind of Science*. Wolfram Media Inc. Retrieved from www.wolfram-media.com Date retrieved, November 2011.

Wong, S. S. Y. and Chan, K. C. C. (2009). EvoArch: An evolutionary algorithm for architectural layout design. *Computer-Aided Design 41* (9) (2009), pp. 649-667.

Xiyu, L., Mingxi, T., and Frazer, J. H. (2005). An eco-conscious housing design model based on co-evolution. *Advances in Engineering Software*, *36* (2005) 115–125.

Yuan, B. and Gallagher, M. (2007). Combining Meta-EAs and Racing for Difficult EA Parameter Tuning Tasks. In Lobo, F. J., Lima, C. F., and Michalewicz, Z. (Eds.), *Parameter Setting in Evolutionary Algorithms*. *Series: Studies in Computational Intelligence, Vol. 54*, 2007, XII, Springer-Verlag, Berlin, Heidelberg, pp121-142.

Zhou, M. X. and Feiner, S. K. (2004). Automated visual presentation: from heterogeneous information to coherent visual discourse. *Journal of Intelligent Information Systems, Volume 11*, Number 3, 205-234, DOI: 10.1023/A:1008685907948.

Zhou, M. X. and Ma, S. (2000). Toward applying machine learning to design rule acquisition for automated graphics generation. *AAAI Technical Report SS-00-04*.

Zitzler, E., Laumanns, M., and Bleuler, S. (2004). A tutorial on evolutionary multiobjective optimization. In Gandibleux, X., Sevaux, M., Sörensen, K., and T'kindt, V. (Eds.), *Metaheuristics for Multiobjective Optimisation*. *Lecture Notes in Economics and Mathematical Systems, Volume 535*, 2004, pp 3-37.

APPENDICES

APPENDIX A: Representation of the Target Buildings

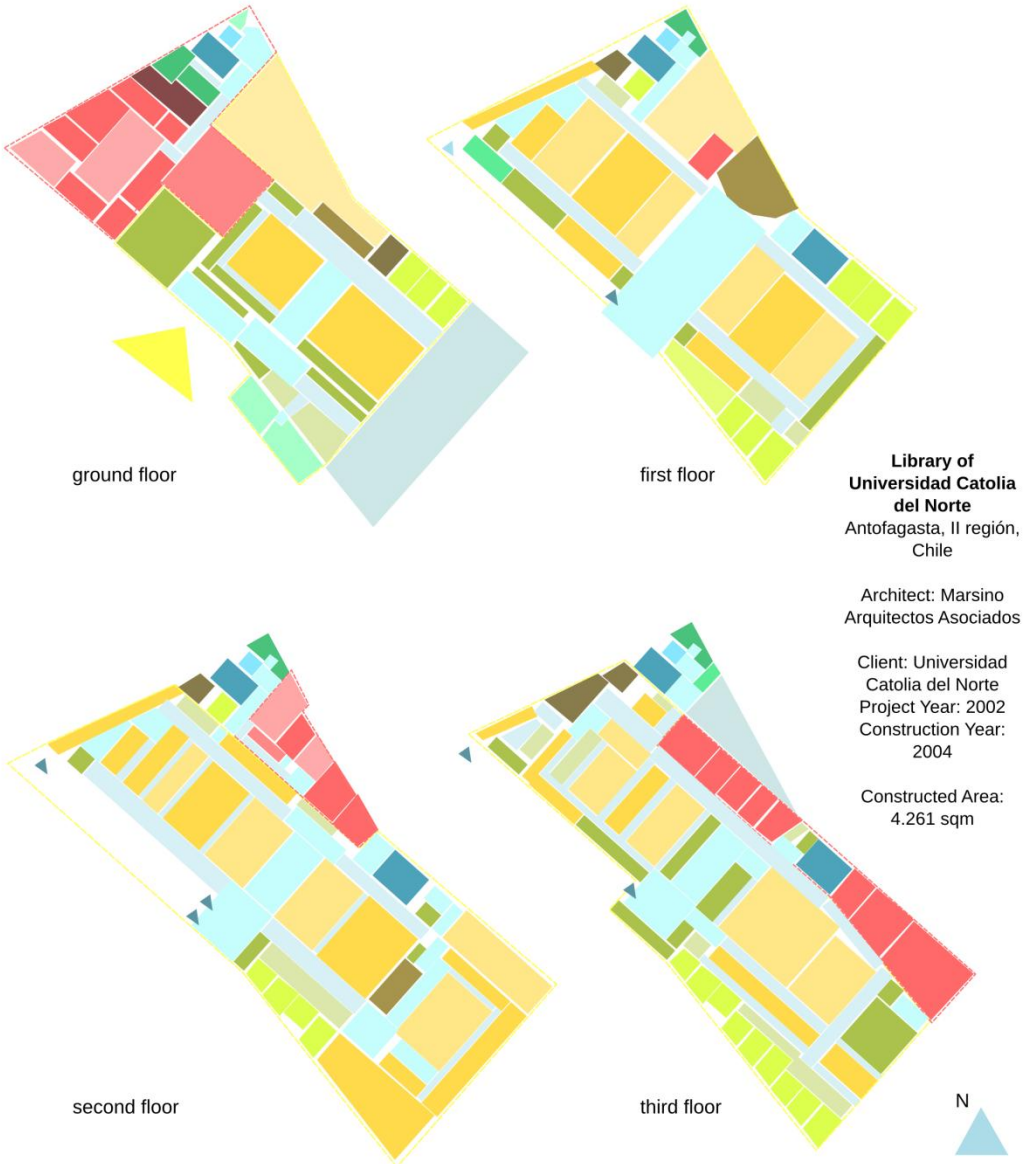


Figure A.1 : Library of "Universitad Catolia del Norte".

Figure A.2 : Library for the Faculty of Philology, Free University of Berlin.

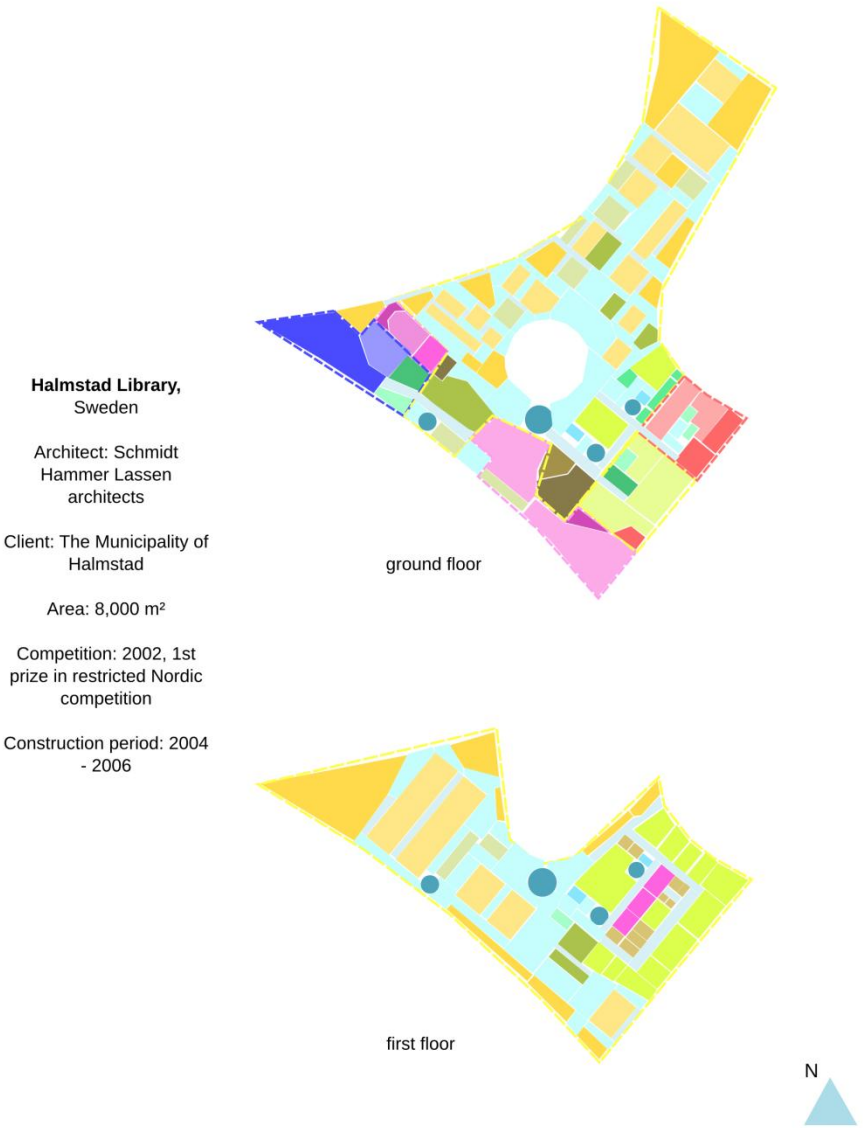


Figure A.3 : Halmstad Library.

Figure A.4 : Santa Monica Library.

Figure A.5 : Troyes Mediatheque.

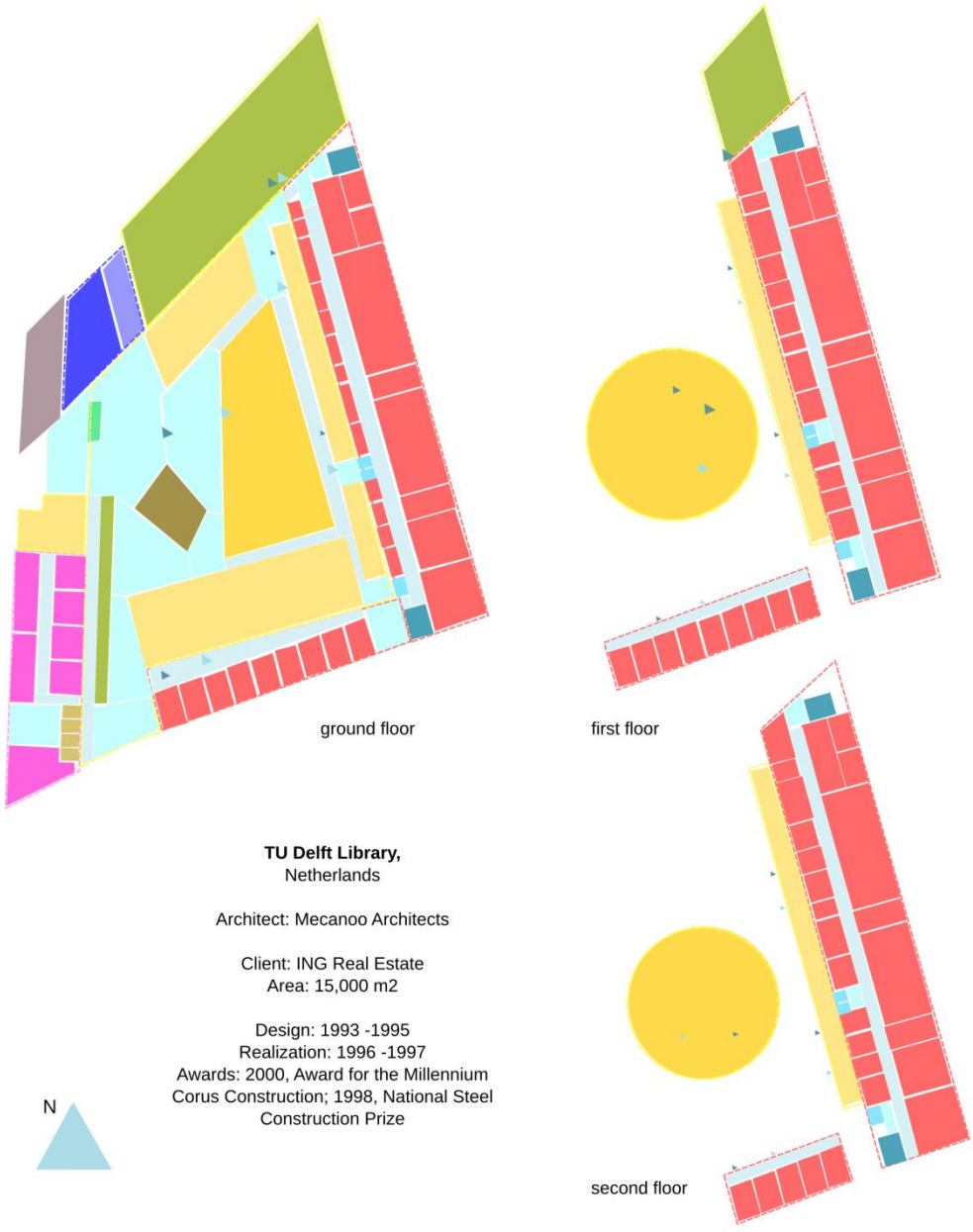


Figure A.6 : TU Delft Library.

CURRICULUM VITAE

Name Surname: N. Onur SÖNMEZ

Place and Date of Birth: İstanbul, 1978

E-Mail: onursonmezn@yahoo.com

EDUCATION:

B.Sc.: Istanbul Technical University, Department of Architecture

M.Sc.: Istanbul Technical University, Institute of Science and Technology, Architectural Design Graduate Program

PROFESSIONAL EXPERIENCE:

- Research assistant, (from 2004 on) İTÜ Faculty of Architecture, Dept. of Arch.

PUBLICATIONS ON THE THESIS:

- **Sönmez, N. O.,** Erdem, A., 2014: Design games: A conceptual framework for dynamic evolutionary design. *A|Z ITU Journal of the Faculty of Architecture, 11 (1)*.
- **Sönmez, N. O.**, Erdem, A., Sarıyıldız, S., 2010: Automated Evaluation and Generation of Graphic Arrangements through Adaptive Evolution. *Generative Art 2010,* Milano.
- **Sönmez, N. O.**, Erdem, A., 2009: Design games as a framework for design, and corresponding system of design games. *ECAADE 2009*, İstanbul.

OTHER PUBLICATIONS:

- Sökmenoğlu, A., **Sönmez, N. O.**, 2013: Exploring reciprocal relationships of land-uses in a historical mixed-use quarter of Istanbul, Measuring mixed-use patterns of Cihangir. *eCAADe 2013*, Delft, the Netherlands.
- **Sönmez, N. O.**, Türkkan, S., Kürtüncü, B., 2012: Layers of designerly knowledge: A collaboration between a design studio and a school for the blind. *Theory by Design 2012,* Antwerp.
- Türkkan, S., **Sönmez, N. O.**, Kürtüncü, B., 2012: Neither individual, nor group: A first year design studio experiment. *ACSA 2012*, Barcelona.
- **Sönmez, N. O.**, 2011: An Architecture for Dissent: A Critical Reading of the New Babylon. *Theory for the sake of theory, ARCHTHEO '11*, İstanbul.
- **Sönmez, N. O.**, Tekin, İ., Üngür, E., 2011: Thinking by design: design as a tool for idea development. *Theory for the sake of theory, ARCHTHEO '11*, İstanbul.