

ARCHITECTURAL VIEWPOINTS FOR GLOBAL SOFTWARE DEVELOPMENT

A THESIS
SUBMITTED TO THE DEPARTMENT OF COMPUTER ENGINEERING
AND THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE
OF BİLKENT UNIVERSITY
IN PARTIAL FULLFILMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

By
Buğra Mehmet Yıldız

August 2011

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Asst. Prof. Dr. Bedir Tekinerdoğan (Supervisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Asst. Prof. Dr. Uluç Saranlı

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Asst. Prof. Dr. Pınar Şenkul

Approved for the Graduate School of Engineering and Science:

Prof. Dr. Levent Onural
Director of Graduate School of Engineering and Science

ABSTRACT

ARCHITECTURAL VIEWPOINTS FOR GLOBAL SOFTWARE DEVELOPMENT

Buğra Mehmet Yıldız
M.S. in Computer Engineering
Supervisor: Asst. Prof. Dr. Bedir Tekinerdoğan

August 2011

Current trends in software engineering show that large software projects have to operate with teams that are working in different locations. The reason behind this globalization of software development stems from clear business goals such as reducing cost of development, solving local IT skills shortage, and supporting outsourcing and offshoring. There is ample reason that these factors will be even stronger in the future, and as such we will face a further globalization of software development. To cope with these problems, the concept of Global Software Development (GSD) is introduced. GSD is a relatively new concept in software development that can be considered as the coordinated activity of software development that is not localized and central but geographically distributed.

Designing a proper architecture of GSD is important to meet the requirements for the communication, coordination and control of distributed GSD teams. However, an analysis of the literature on GSD shows that research in this area has been generally focused on social issues focusing on some concerns such as intercultural communication problems and coordination. It is generally accepted that software architecture design plays a fundamental role in coping with the inherent difficulties of the development of large-scale and complex software. Unfortunately, in both GSD and software architecture design communities, the architecture design of GSD systems has not been explicitly addressed.

This study aims to provide a contribution in this context by explicitly focusing on the architecture design of GSD. A common practice is to model and document different architectural views for describing the architecture according to the stakeholders' concerns. Having multiple views helps to separate the concerns and as such support the modeling, understanding, communication and analysis of the software architecture for different stakeholders. Architectural views conform to viewpoints that represent the conventions for constructing and using a view. In this study, we propose seven architectural viewpoints which have been specifically defined for modeling GSD architecture. To define architecture viewpoints, we first describe a general GSD meta-model. The meta-model has been derived after a thorough analysis of the related GSD literature. The meta-model consists of six different parts which form the abstract syntax of the architectural viewpoints. After the meta-model derivation, we also suggest textual and visual concrete syntaxes for the meta-model in order to complete viewpoint definition.

Supporting the architect in deriving architectural views based on the corresponding architectural viewpoints, we present a question framework. The question framework consists of six sets of questions related to the key concerns of a GSD project. Based on the answers given to the questions in this framework, the GSD application architecture can be derived by applying predefined design actions in the question framework. We have developed the tool called Global Architect which implements the question framework. Global Architect takes as input the answers to the provided questions and subsequently generates the textual architecture description of the required viewpoint. On its turn, the textual description is used to generate the visual presentation of the application architecture for the GSD project.

Keywords: Global Software Development, Architecture Modeling, Architectural Viewpoint, Question Framework, Model-Driven Development.

ÖZET

KÜRESEL YAZILIM GELİŞTİRME İÇİN MİMARİ BAKIŞ AÇILARI

Buğra Mehmet Yıldız
Bilgisayar Mühendisliği Bölümü Yüksek Lisans
Tez Yöneticisi: Y. Doç. Dr. Bedir Tekinerdoğan
Ağustos 2011

Yazılım geliştirme yaklaşımı, son yıllarda tek merkezli geliştirme yerine küresel olarak dağıtılmış geliştirmeye kaymaktadır. *Küresel* veya *dağıtılmış* olarak da adlandırılan bu tip geliştirmede, belli takımlar, coğrafi olarak dünyanın değişik yerlerine dağılmış olan değişik sitelerde çalışır. Bu takımlar yalnızca kodlama üzerine değil, aynı zamanda yazılım pazarlaması, bakım ve test gibi yazılımın diğer alanlarında da çalışmaktadır. Tüm geliştirme, değişik ülkelere dağılmış merkezleri olan bir şirket tarafından idare edilebileceği gibi iş yükü sözleşmeli alt yükleniciler ile de paylaşılabılır. Ticari kaygı gütmeyen açık kaynak geliştirme grupları ve organizasyonlar da Küresel Yazılım Geliştirme (KYG) takımları olarak sınıflandırılabilir.

Dağıtılmış KYG takımlarının iletişim, koordinasyon ve kontrol gereksinimlerini karşılamak için uygun bir KYG mimarisi geliştirmek önemlidir. Bununla birlikte, KYG literatür taraması gösteriyor ki bu alandaki araştırmalar daha çok kültürler arası iletişim ve koordinasyon problemleri gibi sosyal mevzulara yoğunlaşmıştır. Yazılım mimarisinin geniş çaplı ve karmaşık yazılımların geliştirilmesinde önemli bir rol oynadığı bilinen bir gerçektir. Fakat hem KYG hem de yazılım mimarisi toplulukları, KYG mimarisinin tasarımı konusunu açık olarak konu etmemişlerdir.

Bu anlamda, bu çalışma KYG mimarisinin tasarımını konu ederek bu alanda katkı sağlamayı amaçlamaktadır. Mimariyi paydaşların endişelerine göre betimlemek için değişik mimari bakışları modellemek ve dökümlemek genel bir

uygulamadır. Birden çok bakışın kullanılması, farklı paydaşlar için yazılım mimarisinin modellenmesi, anlaşılması, iletişimde kullanılması ve analiz edilmesine yardım etmektedir. Mimari bakışlar, bakışların nasıl inşa edilmesi ve kullanılması gerektiğini tanımlayan bakış açılarına uyarlar. Bu çalışmada, KYG mimarisinin modellenmesi için tanımlanmış olan yedi adet mimari bakış açısı sunuyoruz. Mimari bakış açılarının tanımlanması için, öncelikle genel bir KYG meta-modeli tanımladık. Meta-model ayrıntılı bir literatür araştırması sonucu oluşturuldu. Meta-model, mimari bakış açılarının soyut dilbilgisini oluşturan altı adet parçadan oluşuyor. Meta-modelin tanımlanmasından sonra, meta-model için yazılı ve görsel somut dilbilgisi de, bakış açısı tanımlamasını tamamlamak için önerildi.

Bakış açılarına dayanan mimari bakışların elde edilmesinde mimari desteklemek amacıyla bir soru çatısı sunduk. Soru çatısı, bir KYG projesinin anahtar elementleriyle ilişkili altı adet soru kümesi içeriyor. Bu soru çatısındaki sorulara verilen yanıtlara dayanarak, çatıda önceden tanımlanmış olan tasarım hareketlerinin uygulanmasıyla KYG mimarisi elde edilebilir. Bu soru çatısını *Global Architect (Küresel Mimar)* adını verdiğimiz bir araç geliştirerek uyguladık. Küresel Mimar sorulara verilen yanıtları alarak yazılı biçimdeki mimari tanımlamayı oluşturuyor. Daha sonra, bu yazılı tanımlama kullanılarak KYG projesinin görsel biçimde olan uygulama mimarisi çıkartılıyor.

Anahtar Sözcükler: Küresel Yazılım Geliştirme, Mimari Modelleme, Mimari Bakış Açıları, Soru Çatısı, Model GÜdümlü Geliştirme.

Acknowledgements

It was a great honor for me to work with my supervisor Asst. Prof. Dr. Bedir Tekinerdoğan, who has supported me all the time during my master study. He encouraged me and showed guidance with his sound suggestions and criticisms. His enthusiastic approach towards software engineering made me also excited about studying on this interesting and challenging area.

I would also like to specially thank to Asst. Prof. Dr Pınar Şenkul and Asst. Prof. Dr. Uluç Saranlı for their valuable comments on my thesis.

Members of EA128, Ateş, Damla, Hande and Selen deserve much appreciation for their valuable friendship whole the time we spent together. I must state my special thanks to Saadet for her continuous support.

Finally, I would like to express that this thesis would not have been possible unless my family, my mother Zehra, my father Necati and my younger brother Mert who showed their endless and unconditional love to me. It is my greatest chance in this world to be a part of this great family.

Contents

1. INTRODUCTION	1
1.2 PROBLEM STATEMENT	2
1.3 CONTRIBUTION	3
1.4 OUTLINE OF THESIS	4
2. GLOBAL SOFTWARE DEVELOPMENT	6
2.1 SINGLE SITE DEVELOPMENT	6
2.2 WHAT IS GLOBAL SOFTWARE DEVELOPMENT?	8
2.3 EXPECTED BENEFITS OF GLOBAL SOFTWARE DEVELOPMENT	10
2.4 CHALLENGES OF GLOBAL SOFTWARE DEVELOPMENT	12
3. ARCHITECTURAL VIEWPOINTS AND SOFTWARE LANGUAGE	
ENRINEERING	15
3.1 SOFTWARE ARCHITECTURAL VIEWPOINTS	15
3.1.1 <i>Definitions</i>	16
3.1.2 <i>Current Architectural Frameworks</i>	17
3.2 ARCHITECTURAL VIEWPOINTS AS DSLs	19
4. META-MODEL FOR GLOBAL SOFTWARE DEVELOPMENT.....	21
4.1 DEPLOYMENT UNIT	21
4.2 PROCESS UNIT	23
4.3 DATA UNIT	24
4.4 COMMUNICATION UNIT	26
4.5 TOOL UNIT.....	27
4.6 MIGRATION UNIT.....	28
5. ARCHITECTURAL VIEWPOINTS FOR GLOBAL SOFTWARE	
DEVELOPMENT.....	29
5.1 VIEWPOINT DEFINITION GUIDELINE.....	29
5.2 DEPLOYMENT VIEWPOINT	31
5.3 PROCESS VIEWPOINT	33

5.4 COORDINATION VIEWPOINT.....	35
5.5 DATA VIEWPOINT	36
5.6 COMMUNICATION VIEWPOINT	38
5.7 TOOL VIEWPOINT.....	40
5.8 MIGRATION VIEWPOINT.....	42
6. DERIVING APPLICATION ARCHITECTURE USING QUESTION FRAMEWORK	45
6.1 ADOPTED APPROACH FOR DERIVING APPLICATION ARCHITECTURE	45
6.2 OVERVIEW OF THE QUESTION FRAMEWORK	47
6.3 DEPLOYMENT QUESTION SET	49
6.4 PROCESS QUESTION SET	50
6.5 DATA QUESTION SET	52
6.6 COMMUNICATION QUESTION SET	54
6.7 TOOL QUESTION SET	56
6.8 MIGRATION QUESTION SET.....	57
7. TOOL SUPPORT FOR GENERATING APPLICATION ARCHITECTURE	59
7.1 OVERVIEW OF TOOL SUPPORT	59
7.2 GLOBAL ARCHITECT	61
7.3 XTEXT	63
7.3.1 Global Software Deployment Modeling Language (GSDML).....	64
7.3.2 Global Software Process Modeling Language (GSPML).....	66
7.3.3 Global Software Coordination Modeling Language (GSCOML).....	67
7.3.4 Global Software Data Modeling Language (GSDAML)	69
7.3.5 Global Software Communication Modeling Language (GSCML).....	70
7.3.6 Global Software Tool Modeling Language (GSTML)	72
7.3.7 Global Software Migration Modeling Language (GSML)	74
7.4 USING GRAPHICAL MODELING FRAMEWORK TO DEFINE VISUAL REPRESENTATIONS.....	76
8. RELATED WORK.....	81
9. CONCLUSION	85

List of Figures

Figure 2.1 Conceptual Architecture of Single Site Software Development.....	7
Figure 2.2 Conceptual Architecture of Global Software Development	10
Figure 3.1 Conceptual Model for Architectural Concepts based on [31].....	17
Figure 4.1 Deployment Meta-Model Unit.....	22
Figure 4.2 Process Meta-Model Unit	23
Figure 4.3 Data Meta-Model Unit.....	25
Figure 4.4 Communication Meta-Model Unit.....	26
Figure 4.5 Tool Meta-Model Unit.....	28
Figure 4.6 Migration Meta-Model Unit	28
Figure 5.1 Deployment View: Example Case	32
Figure 5.2 Process View: Example Case.....	34
Figure 5.3 Coordination View: Example Case.....	36
Figure 5.4 Data View: Example Case	38
Figure 5.5 Communication View: Example Case	40
Figure 5.6 Tool View: Example Case	42
Figure 5.7 Migration View: Example Case.....	44
Figure 6.1 Adopted Approach for Deriving Application Architecture	46
Figure 6.2 Question Framework Concepts Model	47
Figure 7.1 Tool Support Activities for Deriving Application Architecture	60
Figure 7.2 Screenshot from Global Architect	61
Figure 7.3 Global Software Deployment Modeling Language	64
Figure 7.4 GSDML: Example Case.....	65
Figure 7.5 Global Software Process Modeling Language.....	66
Figure 7.6 GSPML: Example Case	67
Figure 7.7 Global Software Coordination Modeling Language.....	68
Figure 7.8 GSCOML: Example Case.....	68
Figure 7.9 Global Software Data Modeling Language	69
Figure 7.10 GSDAML: Example Case.....	70
Figure 7.11 Global Software Communication Modeling Language	71
Figure 7.12 GSCML: Example Case.....	72
Figure 7.13 Global Software Tool Modeling Language	73
Figure 7.14 GSTML: Example Case	74
Figure 7.15 Global Software Migration Modeling Language	75
Figure 7.16 GSMML: Example Case	76
Figure 7.17 Annotated Textual ECore Model of Migration Viewpoint.....	78
Figure 7.18 Migration Viewpoint Graphical Editor: Example Case.....	79

List of Tables

Table 2.1 Software Engineer Salaries of Some GSD Participant Countries [16]	11
Table 5.1 Viewpoint Guide Template For GSD.....	30
Table 5.2 Deployment Viewpoint	31
Table 5.3 Process Viewpoint.....	33
Table 5.4 Coordination Viewpoint	35
Table 5.5 Data Viewpoint	37
Table 5.6 Communication Viewpoint	39
Table 5.7 Tool Viewpoint	41
Table 5.8 Migration Viewpoint	43
Table 6.1 Deployment Question Set.....	50
Table 6.2 Process Question Set	52
Table 6.3 Data Question Set.....	54
Table 6.4 Communication Question Set.....	55
Table 6.5 Tool Question Set.....	57
Table 6.6 Migration Question Set	58

Chapter 1

Introduction

Current trends in software engineering show that large software projects have to operate with teams that are working in different locations. The reason behind this globalization of software development stems from clear business goals such as reducing cost of development, solving local IT skills shortage, and supporting outsourcing and offshoring [21]. There is ample reason that these factors will be even stronger in the future, and as such we will face a further globalization of software development [12]. To cope with these problems, the concept of global software development (GSD) is introduced. GSD is a relatively new concept in software development that can be considered as the coordinated activity of software development that is not localized and central but geographically distributed. Typically, GSD teams work on different areas of software business, such as business/marketing, maintenance, testing as well as on software implementation. The development activities may be managed by just one company that is distributed to many centers in different countries or many companies can work on specific projects as organizers and subcontractors for sharing work weight. Open source development groups and organizations that usually do not drive any commercial benefit can also be classified as Global Software Development (GSD) teams [51].

1.2 Problem Statement

A GSD architecture usually consists of several nodes, or sites, on which different teams are working to develop a part of the system. The teams could include development teams, testing teams, management teams, etc. Usually each site will also be responsible for following a particular process. In addition, each site might have its own local data storage.

Overall we can identify three important key concerns in designing GSD:

Communication: Communication mechanisms within and across sites. Typically the different sites need to adopt a common communication protocol.

Coordination: Coordination of the activities within and across sites to develop the software according to the requirements. Coordination will be necessary to align the workflows and schedules of the different sites. An important goal could be to optimize the development using appropriate coordination mechanisms.

Control: Systematic control mechanisms for analyzing, monitoring and guiding the development activities. This does not only include controlling whether the functional requirements are performed but also which and to what extent quality requirements are addressed.

In fact each of these concerns requires further in-depth investigation and has also been broadly discussed in the GSD community. The important issue that we would like to address is that each of these concerns and the way they are allocated in the GSD environment will have a direct impact on the GSD architecture. Likewise, designing a proper architecture for GSD is important to satisfy the requirements for the communication, coordination and control of distributed GSD teams.

It is generally accepted that software architecture design plays a fundamental role in coping with the inherent difficulties of the development of large-scale and complex software. Research on architecture design in the last two decades has resulted in different useful techniques and approaches. Different architectural modeling approaches for representing multiple views of the architecture have been proposed. Yet, designing robust software architectures remains difficult due to the various concerns, and the higher abstraction level. In the software architecture design community the endeavor of software architecting seems to have been mainly focused on architecting in single systems. However, current trends in software engineering show that large software projects have to operate with teams that are working in different locations.

On the other hand, the global software development community seems to have largely discussed non technical issues such as intercultural communication problems and coordination. Unfortunately, less focus has been given from a technical architecting perspective.

1.3 Contribution

This thesis aims to address problems defined in the previous section by explicitly focusing on the software architecture design of GSD. The contribution of this thesis can be summarized as follows:

Meta-model for GSD: For the derivation of the application architecture, we have defined a general GSD meta-model which contains the key concepts of distributed development. The meta-model has been derived after a thorough analysis of the related GSD architecture. The meta-model consists of six different parts which form the abstract syntax of the architectural viewpoints.

Viewpoints for GSD: Based on the defined meta-model, we have described seven architectural viewpoints which have been specifically defined for modeling GSD application architecture. These seven viewpoints help to separate the GSD concerns and as such support the modeling, understanding, communication and analysis of the GSD architecture.

Domain-Specific Language for GSD Application Architecture: Each viewpoint for GSD application architecture includes textual concrete syntax definition which forms domain-specific languages for GSD application architecture. To the best of our knowledge, these are the first DSLs on GSD in the literature.

Question Framework: For helping the architect to derive GSD architecture using viewpoints, we present a question framework. The question framework targets the key concerns of a GSD project. It consists of six question sets each of which has around twenty questions. Each question has an expected answer as defined in the literature. Based on the answers given to the questions in this framework, the GSD application architecture of the particular architectural viewpoint can be derived.

Tool Support: We have developed the tool Global Architect that implements the question framework. Global Architect takes as input the answers to the questions and generates the corresponding textual architecture for the required viewpoint. The textual description of the architecture is mapped to the visual presentation of the application architecture of GSD project using XText and Eclipse Graphical Modeling Framework.

1.4 Outline of Thesis

The thesis is organized as follows: Chapter 2 provides a background of Global Software Development and explains the benefits and problems of GSD approach. Chapter 3 presents the concepts related to software language engineering, architectural viewpoints and frameworks. Further it represents the

adopted approach that we have followed for the derivation of application architecture. In Chapter 4, we present the meta-model of Global Software Development. Chapter 5 describes seven viewpoints together with visual concrete syntaxes. The question Framework is given in Chapter 6. Chapter 7 presents the tool support given for the adopted approach. Chapter 8 provides the related work and finally Chapter 9 concludes the thesis.

Chapter 2

Global Software Development

This chapter provides background information on Global Software Development paradigm. Section 2.1 describes the traditional single site development. Section 2.2 provides the introduction to Global Software Development. Section 2.3 describes the expected benefits from the GSD approach. Finally, the challenges of GSD are presented in Section 2.4.

2.1 Single Site Development

Single site development can be qualified as the traditional way of software development that is still followed by local-based software companies. In this kind of development, one or more development teams of a company are located in one location such as company building. The team or people responsible for customer relations sometimes travel to customer sites in need of communication with customer. The conceptual model of single site development is shown in Figure 2.1.

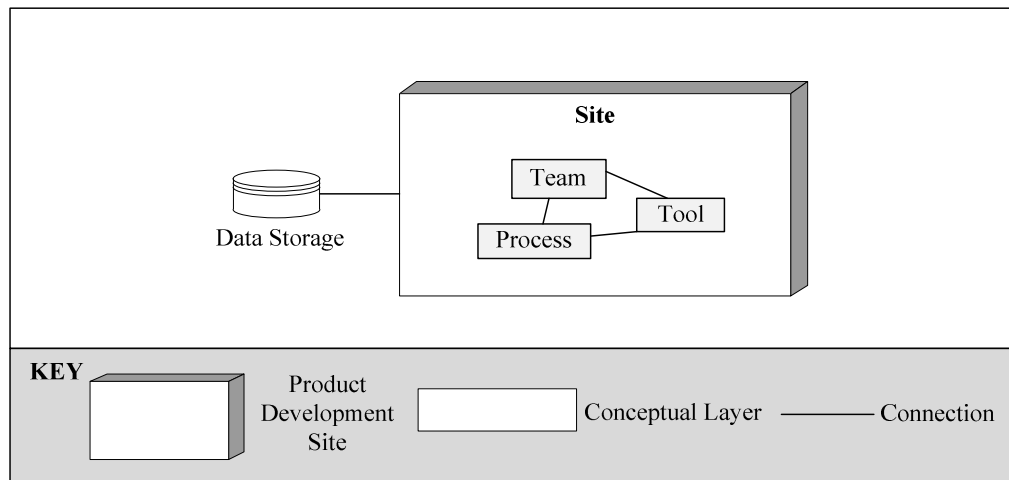


Figure 2.1 Conceptual Architecture of Single Site Software Development

There are some problems faced by the companies that apply single site development approach. Based on the literature study we could derive the following issues:

Long Project Periods: In single site development, teams usually start working at 7-9 am and continue working until 6-7 pm in the local time of the located site. This corresponds to about 8-9 hours of working per day and about 45 hours of working in a week. With this working schedule, the projects last several months or even several years depending on the project sizes.

High Production Cost in Developed Countries: Teams are located constantly in a particular city in single site development. This situation leads to a fixed amount of living expenses for the people working for the company since food and housing costs do not much differ in city base. The average living expenses in a city affect the cost of the developed software: If the city's food and/or housing prices are high like Tokyo, Moscow, Geneva, Zurich or New York [14], the expected cost of the software would be high if the single site development approach is followed. On the other hand, as it can be expected, the second and third world countries offer cheaper living that would lead to cheaper software

development cost (Note that, we compare cost of living versus cost of developed software here; the quality is not mentioned here, one may expect better products in developed countries whose living cost is higher).

Limited Market: Single site development limits the target market of the company. The company generally serves to the customers from the same city or from the cities in the same country because of high travel and communication costs to other countries. The companies that apply single site development approach rarely find customers from other countries that stop these companies to take a pay from offshore markets. So, the company cannot get into a competition with other companies in different markets.

Lack of Expertise: Another limitation is that sometimes, single site development suffers from lack of expertise. This is because the single site development approach depends on the local skill pool. If a company needs an expert from a particular area, it has to find a person located near to the company's location. If there is a lack of expertise in the available skill pool, the company may choose to train its own workers or hire/employ a person from other locations or abroad where both approaches are costly compared to the local solution.

Governmental Obligations: A company who works in single site should obey the regulations and laws put by the government of that country. Sometimes these regulations can put boundaries to software development area or to market competitions and the company cannot escape from these regulations unless the company has other branches in other countries.

2.2 What is Global Software Development?

With the act of globalization, the companies working in many different areas have moved their production centers to other countries for getting a higher share from the world pie especially in the last two decades. The software industry is

also affected with world globalization trend. In order to solve the problems of single site development that we stated in the previous section and get advantage of outsourcing (hiring a subcontractor in order to do some function) and offshoring (locating some functions to be done in another country), many software companies have carried their development sites to other countries such as India and China.

In this so-called global or distributed [11] software development, there are a number of teams working on different sites that are distributed geographically across the world. These teams work not only coding but also focus on different software development jobs such as business/marketing, maintenance, testing. Development of a project may be managed by just one company whose centers are distributed to different countries or subcontractors around the world can share work weight of organizer companies. Not only private commercial companies but also open source development groups and organizations that usually do not drive any commercial benefit can be also classified as Global Software Development (GSD) teams. Conceptual architecture of GSD is given in Figure 2.2.

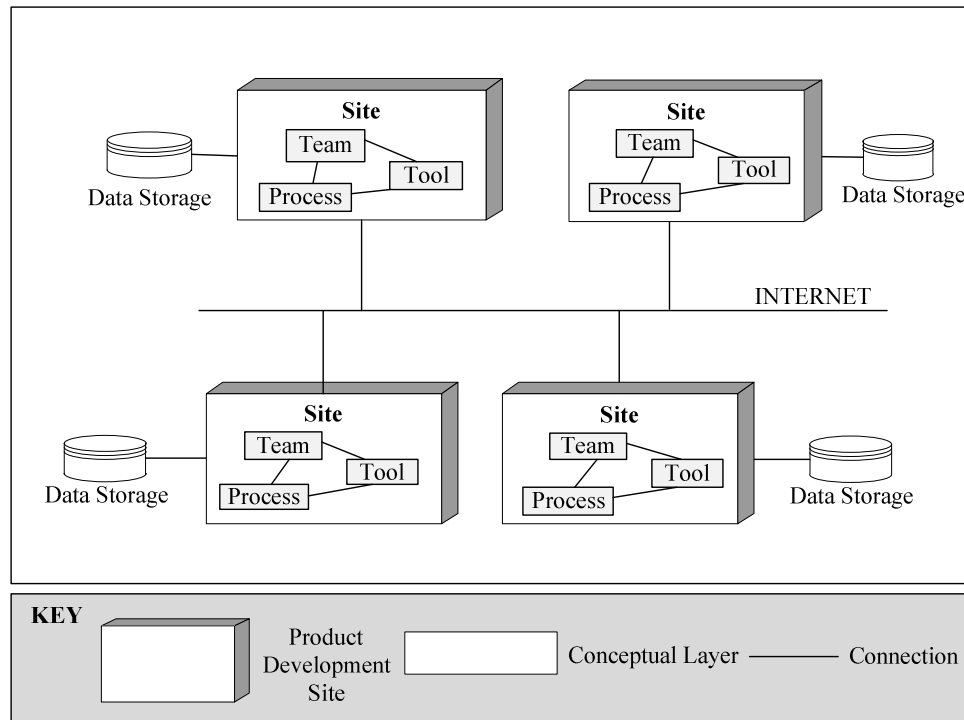


Figure 2.2 Conceptual Architecture of Global Software Development

2.3 Expected Benefits of Global Software Development

The main purpose of applying Global Software Development approach for a company is to gain more profit from the development. To reach this target, GSD offers the following advantages:

Cheaper Production: Decreasing the cost of the production is generally considered as one of the main motivations for adopting GSD. One of the main factors for the cost of production is the wages of the engineers in the project. By moving the production centers to the countries in which is cheaper than the company's home country that the cost for wages and likewise cost of production can be substantially decreased [15]. To show the wage difference between the GSD participant countries, Table 2.1 shows lower bound yearly incomes of software engineers in different countries [16]. The reason why most of the

American outsourcing is done to India, the country with the lowest average minimum salary is clear from this table.

Country	Minimum Yearly Salary(\$)
United States	42,757
India	6,784
China	10,180
Ireland	50,588
Russia	12,825
Singapore	15,534
New Zealand	33,566
Australia	41,961
Germany	32,002
Netherlands	35,247
England	37,669
Brazil	15,428
Turkey	20,793

Table 2.1 Software Engineer Salaries of Some GSD Participant Countries [16]

Large Pool of Skilled Labor: In single site development, a company depends on the local skill pool. On the contrary, globally distributed companies can reach to different countries' man power. When an expertise is needed on some subject, it is often easier to solve this by using globally distributed centers rather than one center located at one country [17] [18].

Faster Time to Market: One of the main advantages of GSD is that it supports "Follow-the-sun" approach [19]. As it can be understood from its name, the daily development time may increase from 8-9 hours up to 24 hours. This approach takes the advantage of the sun's daily action on earth and so different time zones. As one center finishes its daily shift, the other center in another place in the world can just start its daily shift and so 24 hours of the day is used for development.

New Markets: By allocating different centers in different countries, a company catches the chance of joining the local markets. So, the company can reach more customers. In [20], it is stated that new job opportunities created locally can lead to find more customers in that local area.

Other advantages: Other than these commonly accepted benefits of GSD, the study in [21] reports some unknown benefits of GSD. This study reports additional benefits under organizational, team and process/task categories. From organizational perspective, since a company can reach to a large pool of skilled labor, the company's innovation ability and resource allocation improves. From teams perspective, task modularization improves because tasks are tend to be assigned in a way that interdependencies are kept low and so the coordination cost decreases. GSD requires keeping track of the information flow so more formal and explicit processes are defined and documentations and communication histories are recorded.

2.4 Challenges of Global Software Development

Although Global Software Development has several clear benefits for a company, it also has to face important challenges. In this context, we do not mean problems that are also common to single system development but indicate problems specific to Global Software Development settings where several teams located in different parts of the world. The following challenges have been identified in the GSD literature:

Synchronous communication: From the nature of the distributed environment, communication between sites becomes a vital activity for developers. But, at the same time difficulties exist in side-to-side communications. One of the difficulties is about synchronous communication. Each site has its own local time zone and because of the time zone differences, it becomes hard to find overlapping hours in the shifts. In [22], it is stated that finding common times for collaboration can cause delays in feedback.

Informal communication: Another problem is about informal communication activities which occurs in a unplanned way between developers such as phone

calls made in case of information needs or chats done during breaks. In unstable environments and projects that are open to changes, informal communication is especially important [23] [24]. But because of the distance, informal communication sometimes becomes harder or even impossible. The lack of informal communication leads to several negative situations such as absence of trust [25] [26] and to lose team spirit. This creates an obstacle in front of information sharing and communication tendency.

Teams' background: Teams in different sites come from different cultures: both social and work culture. Most of the time, they haven't worked together before, they come from different trainings, and they don't know about each others' habits and thoughts. As social culture impact, one's action can be felt as rude by the opposite site. For teams that are located in separate countries, language appears as a disadvantage sometimes. So, background difference can cause misunderstandings and misinterpretations which stops successful information exchange.

Awareness: Because of these communication problems, and since teams at separate sites share little content in common, they usually don't know what the people in the other sites are doing. This lack of awareness of other sites' activities may introduce delays and unnecessary reworks.

Interdependencies between sites: As a general practice, task allocation is done in a way that interdependencies between sites are as small as possible so that coordination and communication problems are reduced. Task allocation based on software architecture creates a challenge for software engineers to derive a suitable architecture and work division for multi-site development. Not only software architecture, but also dependencies based on other project artifacts such as detailed design, code, test and process documents causes more sites to collaborate. But it is not possible to do this ideally where no intersection work

exists. So, more coordination and communication mechanisms are needed to get rid of extra delays [27].

Interoperability: GSD sites often differ in terms of work culture so they use different tools and development platforms. When activities that need collaboration of sites are done, interoperability becomes a barrier if sites don't use similar platforms or tools.

Architectural Design: All the above challenges have been somehow discussed in the GSD literature. Unfortunately, as stated before, the notion of architecture design has not yet been explicitly addressed. This includes architecture design, the modeling and documentation of the architecture, architecture analysis and architecture realization. In this thesis, we mainly focus on architecture design and its documentation.

Chapter 3

Architectural Viewpoints and Software Language Engineering

This chapter presents information about software architectural viewpoints and our approach for deriving application architecture. Section 3.1 gives some background on software architectural viewpoints; it starts with definition subsection and ends with some example architectural frameworks from the literature. Section 3.2 expresses architectural viewpoints from domain-specific language perspective. The chapter ends with adopted approach for deriving application architecture of Global Software Development.

3.1 Software Architectural Viewpoints

A stakeholder can be defined as a person or a group of people who have some interest in the target system. Each stakeholder has some interest, expectations or concerns regarding to the system and the architect needs to satisfy the concerns of stakeholders. In order to satisfy the stakeholder concerns, looking at the system from just one perspective is not suitable. So, a common practice is to model and document the architecture from different viewpoints targeting

different stakeholders [1] [28] [29] [30]. General concepts regarding to architectural viewpoints are explained in the following subsection.

3.1.1 Definitions

IEEE provides the fundamental definitions related to software-intensive systems in IEEE 1471 standard [31]. In this standard, the following definitions are given:

System: A collection of components organized to accomplish a specific function or set of functions.

System Stakeholder: An individual, team, or organization (or classes thereof) with interests in, or concerns relative to, a system.

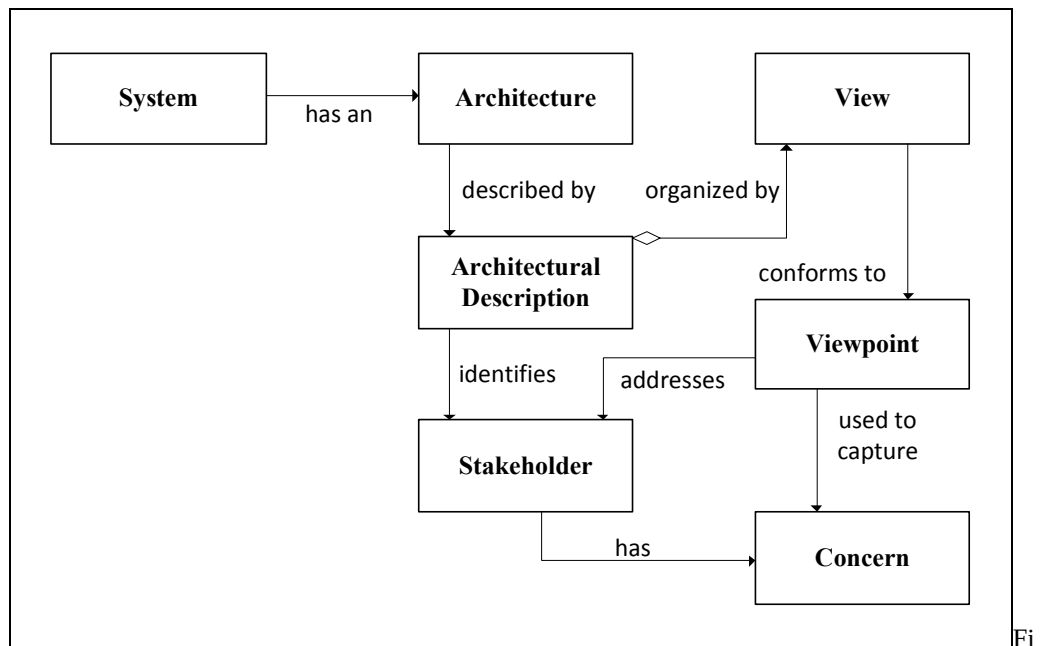
Architecture: The fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution.

Architectural Description: A collection of products to document architecture.

View: A representation of a whole system from the perspective of a related set of concerns.

Viewpoint: A specification of the conventions for constructing and using a view. A pattern or template from which to develop individual views by establishing the purposes and audience for a view and the techniques for its creation and analysis.

Moving from these definitions, the conceptual model given in Figure 3.1 can be derived.



Every system has an architecture. Architecture of a system is described in architectural description. Architectural description consists of several views each of which conforms to a viewpoint. Each viewpoint is defined in order to capture the concerns of stakeholders so addresses different stakeholders.

3.1.2 Current Architectural Frameworks

Architectural frameworks organizes architectural viewpoints and so allow architect to use these viewpoints in a consistent and correct manner. There are different architectural frameworks proposed in the literature both for software intensive and purpose specific use. Some important example architectural frameworks are discussed below.

In the Views and Beyond Approach [1], the authors introduce the notion of architectural styles as specialization of elements and their relations together with usage constraints. Architectural styles are often recurring forms that are used to present different aspects of a software system. The book represents several styles and these styles are grouped under three categories: Module, Component-

and-Connector and Allocation. Module styles are used to show basic units of implementations. Component-and-Connector styles organize system units during execution time. Allocation styles connect software and non-software system units. The application of a style results in a view of the system.

Kruchten's 4+1 Architectural View Model [29] collects the views of the software intensive systems into four views: logical, development, process and physical. End user functionality is focused by logical view. The development view reflects the system from implementation point of view; this view is especially used by software developers. The process view represents dynamic perspective of a system. The physical view concentrates on how the software elements are mapped to non-software units. As the plus one, use cases and scenarios are used for satisfying end user interest.

Siemens's Four View Model [28], consists of the following four viewpoints: conceptual, module, code and execution viewpoint. The Four View Model is similar to the Kruchten's 4+1 Architectural View Model. The conceptual view is mapped to the use case model, the module view and code view correspond to logical and development views respectively; the execution view which represents dynamic side of software systems is similar to process view in 4+1 Model.

In this thesis, we focused on the software architectural frameworks but representing the system from different perspectives approach is also applied to enterprise architectures. Some examples can be shown as The Open Group Architecture Framework (TOGAF) [36], Zachman's Framework for Enterprise Architecture [37] and ISO (ISO/IEC 10746) Reference Model of Open Distributed Processing (RM-ODP) [35].

3.2 Architectural Viewpoints as DSLs

Models are different in nature and quality and different classifications of models have been provided in the literature. Mellor et al. [52] make a distinction between three kinds of models, depending on their level of precision. A model can be considered as a *Sketch*, as a *Blueprint*, or as an *Executable*. According to [52], an executable model is a model that has everything required to produce the desired functionality of a single domain. Executable models are more precise than sketches or blueprints, and can be interpreted by model compilers.

Demirli and Tekinerdogan [32] introduce an approach for defining architectural viewpoints as domain-specific languages. Most architectural viewpoints function as a way of communication between stakeholders as sketches or blueprints for the detailed design stage. The study notes that although recent viewpoints tend to be defined more precisely, still they lack of some precision. This lack of precision may lead to lower the quality of the architectural documentation. So, a software language engineering approach of defining viewpoints as domain-specific languages will enhance the formal precision of architectural viewpoint since it will allow architect to get advantage of DSLs such as defining executable, formal views.

Every DSL has a meta-model. Meta-models can be considered the languages for models. In both software language engineering [34] and model-driven development domains [33], a meta-model should have the following elements:

Abstract syntax: The descriptions of the concepts which forms the vocabulary of the languages and how these concepts come together to create a valid model.

Concrete syntax: The adopted notation for representing the concepts of the language. Basically, we can distinguish between visual concrete syntax and textual concrete syntax. The visual concrete syntax defines how the concepts in

the language should appear in the visual diagrams; textual concrete syntax allows the models to be defined in textual form.

Semantics: Describes meanings of the concepts and relations defined in abstract syntax. Semantics can be described using natural languages.

Moving from these elements of meta-models, the viewpoints can be also considered as languages for defining views [32]. So, the viewpoints can be thought as DSLs. In this study, we evaluate viewpoints of GSD as the languages for defining GSD views. Chapter 4 gives the abstract syntax definitions and Chapter 5 presents the concrete syntax definitions of GSD architectural viewpoints.

Chapter 4

Meta-Model for Global Software Development

Based on the literature of GSD, we have defined a meta-model for GSD that defines the concepts and their relations. The meta-model presented in this section primarily aims to serve as the abstract syntax of the GSD viewpoints. Other than this functionality, the meta-model can also be used for enhancing the understanding of GSD and can support the model transformation. The designer may choose to derive an application architecture by him/herself if s/he doesn't want to use the automated process that we presented in the previous section by using the meta-model directly.

Since the meta-model is quite large, we decomposed it into six meta-model units. Each of these meta-model units includes semantically close entities and address different concerns. Each unit is mapped to corresponding architecture viewpoint. These units are Deployment, Process, Data, Communication, Tool and Migration. We explain each unit in the following sections.

4.1 Deployment Unit

Deployment Unit concerns the deployment of the teams to different sites. The meta-model of this unit is shown in Figure 4.1.

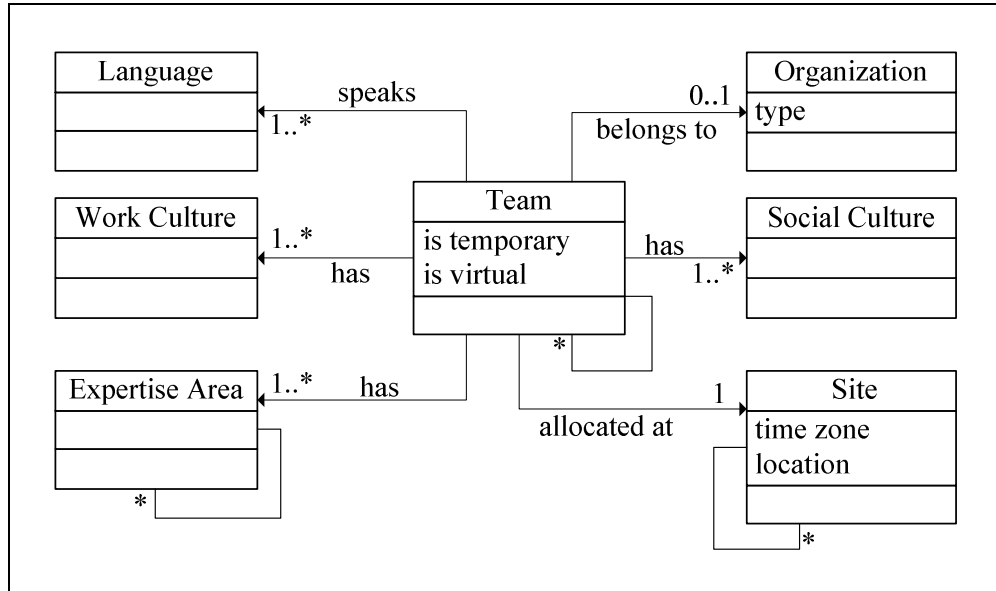


Figure 4.1 Deployment Meta-Model Unit

Team is the primary essential entity in the meta-model and is defined as a group of persons that work together to achieve a particular goal. A *Team* may be organized in a temporary way that it will be dismissed after its function is complete. Each *Team* has particular *Expertise Areas*. *Team* is allocated at a particular *Site*. *Site* maps to a country, city or a building where a *Team* works at. Location attribute determines where *Site* is placed in the world. Time zone shows the local time of *Site*. *Teams* may belong to different types of *Organizations*, such as commercial organizations, subcontractors or non-profitable organizations such as open source communities. *Teams* can be from different countries and depending on the society they are in, they may have different *Social Cultures*. Like *Social Culture*, *Team*'s background including work experience, the time that members work together, their habits are captured by *Work Culture* entity.

Expertise Area, *Team* and *Site* can be further decomposed into sub-parts. For example, a Software Team may consist of sub-Teams each responsible for Design, Implementation, Testing and Integration.

4.2 Process Unit

Process Unit concerns the different kind of processes in GSD. The meta-model of this unit is shown in Figure 4.2.

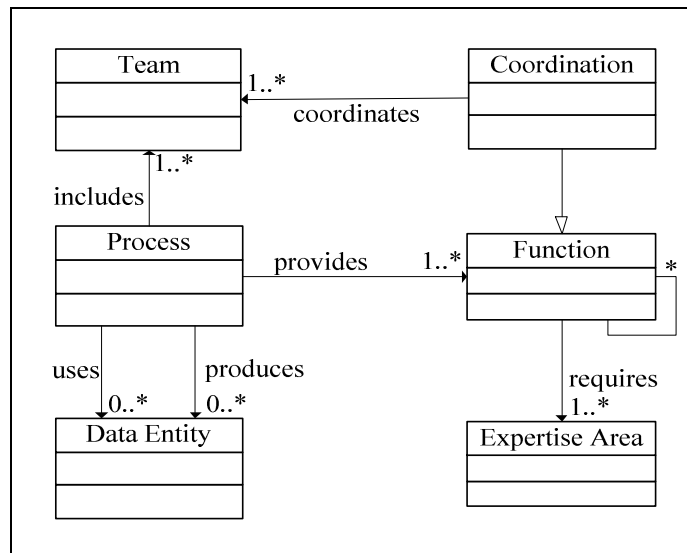


Figure 4.2 Process Meta-Model Unit

Process is defined as a planned set of activities that aims to provide some service. *Teams* participate in *Process* in order to provide some service. Service is defined with *Function*. A *Function* can be any service during software development process that requires some *Expertise Areas* such as software development, architecture design, business management, requirements elicitation and so on. *Coordination* is also a *Function* that should be provided for coordinating several *Teams*' activities. A *Process* consumes or uses several different *Data Entities* and also creates other *Data Entities* for providing targeted *Functions*. For supporting activities defined in *Process*, *Process*

concept is further specialized into Workflow, Business Process and Development Process (not shown in figure).

Process is defined as a single entity in this meta-model unit. But in reality, *Process* is a complex entity which contains a complicated model of different entities such as Roles, Activities, Checkpoints, etc. For the definition of Process itself as a meta-model, several process meta-model definitions are proposed for serving different purposes. Our meta-model can behave interoperable with these process meta-models through the entities such as Team, Data Entity, Expertise Area or Function that are presented in our meta-model. As an example of process meta-model definitions, Object Management Group's (OMG) "Software & Systems Process Engineering Meta-Model Specification" (SPEM) [2] can be considered. Team and Data Entity elements of Process meta-model can be mapped to Role Use and Work Product Use elements in SPEM.

4.3 Data Unit

Data Unit is for representing ownership and physical deployment of software development data. The meta-model is shown in Figure 4.3.

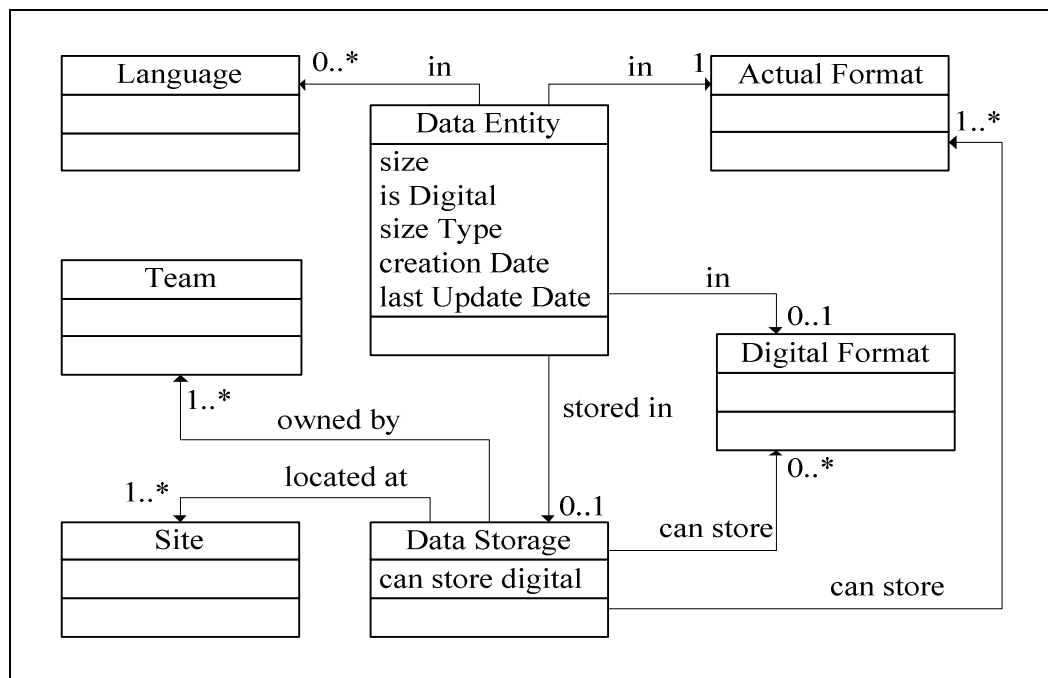


Figure 4.3 Data Meta-Model Unit

Data Entity is the fundamental entity of this meta-model. It represents any piece of data: digital, textual or informal piece of information such as notes taken by developers, telephone calls that are usually not recorded. *Data Entity* has size whose unit is defined by size type; for example, a 120-page report, 6 minutes of voice record, 2 gigabyte of digital data. Creation date and last update date show the history of *Data Entity*. *Data Entity* has *Actual Format* where it can be one of predefined formats (video, sound, text, picture and complex-Data Entity) or some designer defined format. If *Data Entity* is digital, then in addition to *Actual Format*, it has a *Digital Format*. *Data Entity* may be implemented in one or more *Languages*.

Data Entity is stored in *Data Storage*. *Data Storage* corresponds to any object in real world that can store information. For example, some textual document is stored in paper form, or it is stored in a voice record, or it is stored digitally in the format of some text editor. *Data Storage* has ability to store some *Actual Formats* and if it can store digital data, then it can support some *Digital Formats* also. A *Data Storage* instance is owned by one or more *Teams* and it

can be located in one *Site* or may be distributed over several *Sites* like distributed databases.

4.4 Communication Unit

Communication Unit focuses on the representation of both formal and informal communication activities between *Teams*. The meta-model is shown in Figure 4.4.

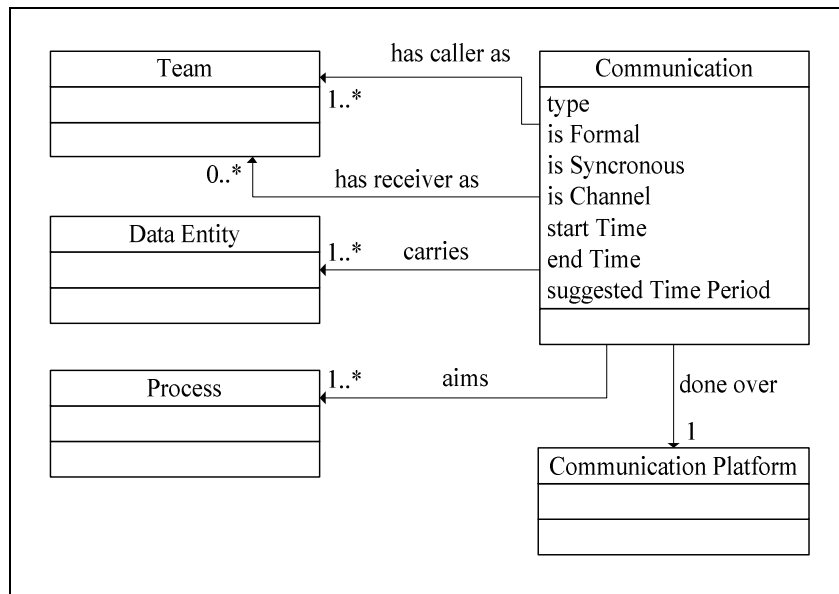


Figure 4.4 Communication Meta-Model Unit

Communication is done over *Communication Platform* in the context of *Process* and it can be an instance of sudden/event based communication activity like a telephone call or a continuous communication channel such as a discussion forum. Type attribute is for representing in which way *Communication* takes place such as email, phone call, face-to-face chat and so on. If *Communication* instance is not a channel, then it means it is sudden communication activity and so it has start and finish time attributes. Suggested time period is an important attribute for GSD since *Teams* work in different time zones, some *Communication* channels such as telephone channel can be used

effectively in a defined time period. For example, phone calls should be done during the hours when both sides are in or around their work hours.

Communication has two sides which are caller and receiver. Generally speaking, caller starts communication and receiver is the one who is called by caller. For example, an email sender is classified as caller and receiver is the one who receives email. Sometimes, there can be multiple callers such as video conferences or there can be multiple receivers such as discussion forums. It is also possible that caller and receiver are the same such as a planned meeting. For all cases, caller and receivers are considered as *Teams* in this meta-model. While *Teams* communicate, one or more *Data Entities* are carried in the context of *Communication*. Speech in telephone calls, electronic mails' body contents or attachments can be shown as examples of *Data Entities* carried by *Communication*.

4.5 Tool Unit

Tool Unit captures details of tools used by *Teams* for communication and providing *Functions*. The meta-model is shown in Figure 4.5.

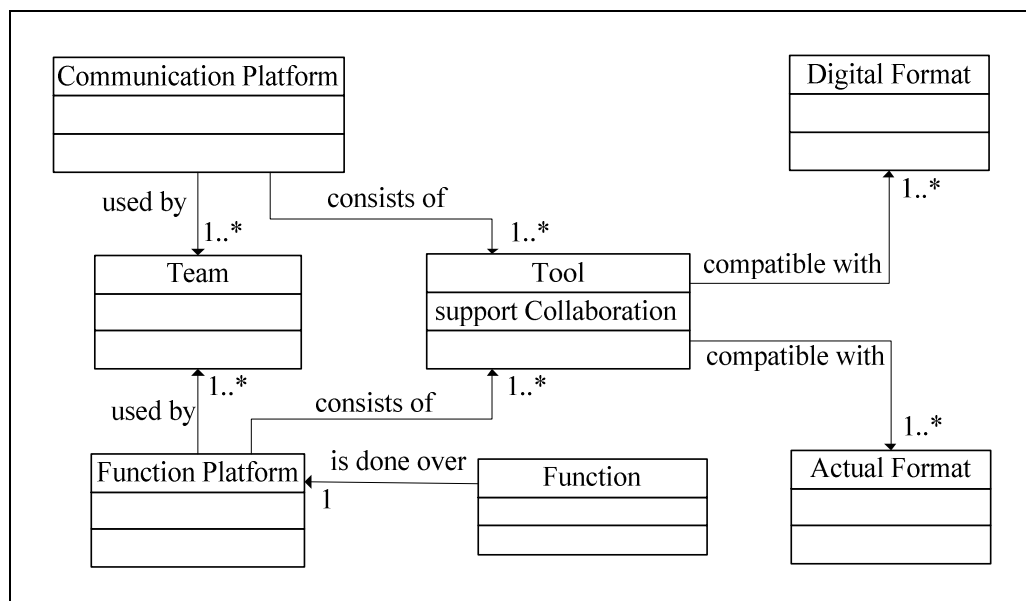


Figure 4.5 Tool Meta-Model Unit

Tool is compatible with one or more *Actual Format* and *Digital Format*. Platform is the set of *Tools* used by *Teams* for communication or providing some functions. Depending on the purpose, the platform is defined as *Function Platform* or *Communication Platform*. Each *Function* is provided over a particular *Function Platform*.

4.6 Migration Unit

Migration Unit concerns the migration and traveling of teams during GSD activities. These travels are especially needed in the first and final phases of the projects to ease and support coordination and integration. The meta-model is shown in Figure 4.6.

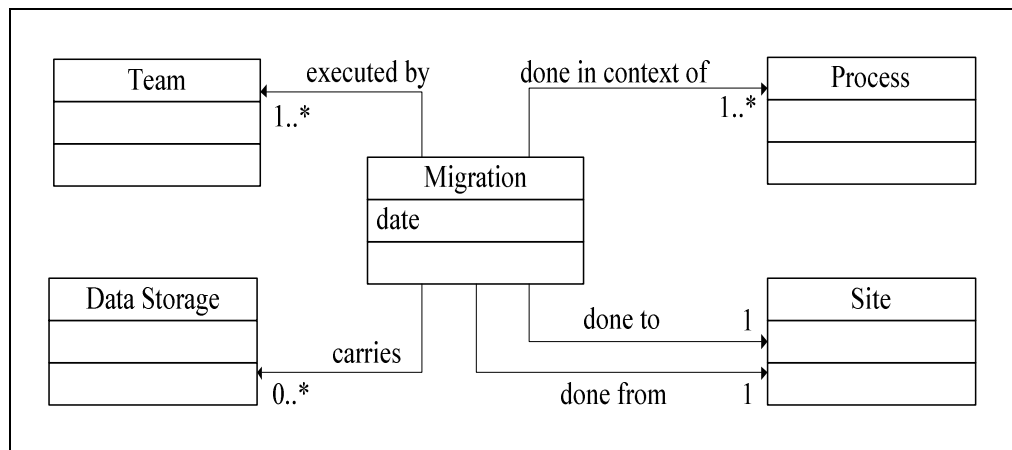


Figure 4.6 Migration Meta-Model Unit

Migration is executed by one or more Teams from Site to Site at a particular date. In a Migration, Teams may carry Data Storage such as documents, digital data containers and so on. Migration is executed in the context of Process.

Chapter 5

Architectural Viewpoints for Global Software Development

Based on the meta-model for GSD that we have defined in the previous chapter, in this chapter we will define the viewpoints for GSD. In Section 5.1, we present the Viewpoint Definition Guideline describing the template for viewpoint definitions. The subsequent sections define the Deployment, Process, Coordination, Data, Communication, Tool, Migration viewpoints. In each of these viewpoint sections, based on the corresponding abstract syntax as defined in the previous chapter, first viewpoint definition is given, followed by an example.

5.1 Viewpoint Definition Guideline

Defining a new architectural viewpoint implies writing a viewpoint guide. This is similar to the notion of style guide as defined in [1]. The viewpoint guide defines the vocabulary of the architectural element and relation types, and defines the rules for how that vocabulary can be used. For defining a viewpoint guide for a particular quality concern we apply the template as defined in Table 5.1.

Viewpoint Element	Description
<i>Name</i>	Unique name for the viewpoint
<i>Element Types</i>	The architectural element types native to the viewpoint
<i>Relation Types</i>	The relation types among architectural elements
<i>Properties of Elements</i>	Additional information on the element types
<i>Topology Constraints</i>	The rules of composition of the elements and relations.
<i>Notation</i>	The adopted notation for the element types and relation types. The notation can be textual or visual.
<i>Relation to other views/viewpoints</i>	The relation to other viewpoints other than the base viewpoint

Table 5.1 Viewpoint Guide Template For GSD

A viewpoint defines the template for the views that can be instantiated from it. In that sense, we consider viewpoints as meta-models representing the basic concepts of the architecture from a particular perspective. To define viewpoints and likewise the viewpoint template, we believe it is necessary to define the corresponding meta-model. As we mentioned before given the key elements of a language, which are abstract syntax, concrete syntax and semantics, we can also evaluate viewpoints of Global Software Development as the languages for defining GSD views.

Likewise to define the viewpoints for GSD, we have to define the corresponding language including both the abstract syntax and the concrete syntax. The abstract syntax is defined based on the literature of GSD under the name of GSD meta-model in the previous chapter. Concrete syntax is included in the viewpoint guide of each viewpoint. In this study, we presented seven different parts each representing an architectural viewpoint for GSD systems. The viewpoints that we have defined based on the meta-model units in the previous section are Deployment, Process, Coordination, Data, Communication, Tool and Migration Viewpoints. These viewpoints are described in more detail in the following subsections.

5.2 Deployment Viewpoint

Deployment Viewpoint aims the representation of deployment of project teams to different sites around the world. Guide for this viewpoint is given in Table 5.2.

Viewpoint Element	Description
<i>Name</i>	Deployment Viewpoint
<i>Element Types</i>	Team, Site, Organization, Language, Expertise Area, Social Culture, Work Culture
<i>Relation Types</i>	allocated-at, belongs-to, has, speaks, parent-child
<i>Properties of Elements</i>	All elements: name, description, id Team: isTemporary, isVirtual Site: timezone, location Organization: type
<i>Topology Constraints</i>	- Team hierarchy must be in the structure of tree. - Expertise Area hierarchy must be in the structure of tree. - Site hierarchy must be in the structure of tree.
<i>Notation</i>	Site:  <Site name> <Teams> Team:  <Organization name> <Team name> <Properties> Language: as property of Team Expertise Area: as property of Team Social Culture: as property of Team Work Culture: as property of Team
<i>Relation to other views/viewpoints</i>	- Team entity is common for all viewpoints. - Expertise Area entity is used in Process viewpoint.

Table 5.2 Deployment Viewpoint

As an example case, consider a GSD project with 7 Sites located at different places in the world. Company A, the owner of the project, has its center operating in United States. Requirement Analysis is done in New York and Architecture Team works in Silicon Valley, California. Company A has development center in New Delhi, India where software development and test Teams work. Also, Company A works with a subcontractor company, Company B for some particular components. Company B's center is located in Sao Paulo,

Brazil. Company B also has development and test Teams. Figure 5.1 shows the Deployment View of the case given in the description. The Deployment View is derived following the Deployment Viewpoint definition given previously.

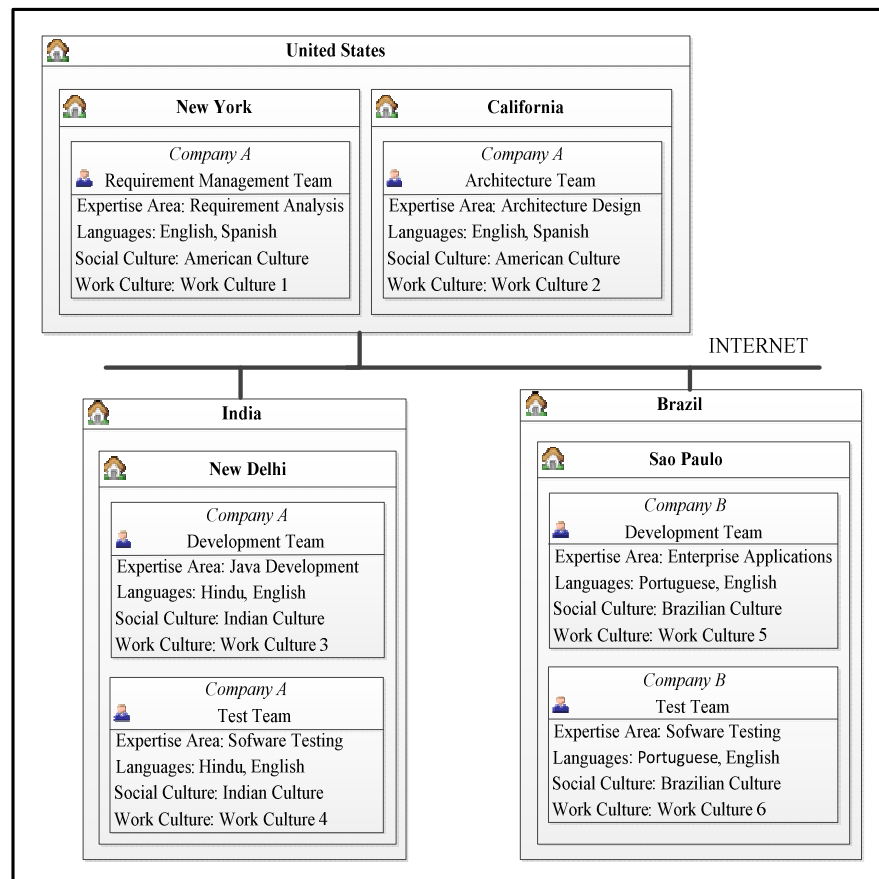


Figure 5.1 Deployment View: Example Case

Team definition is the starting point of all viewpoints. The reason of this is that Team entity is included in all meta-model units. Since Team is a key entity, we use the teams defined in Deployment Viewpoint example as a base for the view definitions in the following viewpoint sections.

5.3 Process Viewpoint

Process Viewpoint aims the representation of the processes that are adopted by different teams in the GSD project. The guide for this viewpoint is given in Table 5.3.

Viewpoint Element	Description
<i>Name</i>	Process Viewpoint
<i>Element Types</i>	Team, Coordination, Process, Function, Data Entity, Expertise Area
<i>Relation Types</i>	coordinates, includes, provides, uses, produces, requires, parent-child, extends
<i>Properties of Elements</i>	All elements: name, description, id
<i>Topology Constraints</i>	- Function hierarchy must be in the structure of tree.
<i>Notation</i>	Data Entity:  <Data Entity name> Team:  <Team name> Function:  <Function name> <Expertise Areas> Process:  <Process name> <Teams> Function-Process association: <Function> → <Process> Data Entity-Process association: <Used Data Entity> → <Process> <Process> → <Produced Data Entity> Expertise Area: as property of Function
<i>Relation to other views/viewpoints</i>	- Team entity is common for all viewpoints. - Expertise Area entity is used in Deployment viewpoint. - Data Entity entity is used in Data viewpoint.

Table 5.3 Process Viewpoint

Consider the following as an example case based on team definitions in the previous section's example: Requirement Management Team follows a Requirement Analysis process which is done for serving Requirement Derivation function. The output of the Requirement Analysis process is Requirement Description Document. This document is used by Architecture

Team in order to design architecture. The document is an input to Architecture Design process and this process creates Architecture Description Document which then be used in implementation process. Since there are two Sites taking role in development, Architecture Description Document is an input to two different development processes whose outputs are working software units. This process flow is shown in Process View in Figure 5.2. The Process View is derived following the Process Viewpoint definition given before.

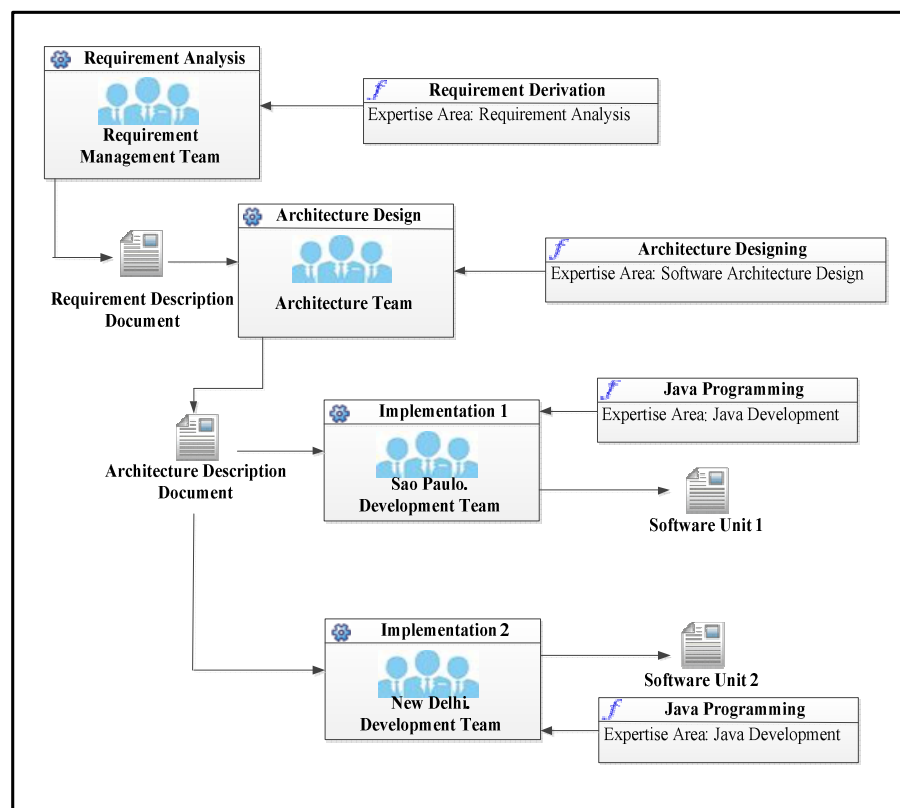


Figure 5.2 Process View: Example Case

For keeping the diagram simple, we didn't include some other details of this process flow such as Detailed Design stage coming after Architecture Derivation before starting implementation or Testing of software units that are gotten at the end of Implementation processes.

5.4 Coordination Viewpoint

Coordination Viewpoint captures the coordination activities of project teams. There is no separate meta-model unit for this viewpoint, but it uses Process meta-model unit as the abstract syntax. The viewpoint aims the representation of the coordinated and coordinator teams. The viewpoint guide is shown in Table 5.4.

Viewpoint Element	Description
<i>Name</i>	Coordination Viewpoint
<i>Element Types</i>	Team, Coordination
<i>Relation Types</i>	coordinates, includes
<i>Properties of Elements</i>	All elements: name, description, id
<i>Topology Constraints</i>	- Coordinated and coordinator Teams must be different.
<i>Notation</i>	Coordination:  <Coordination name> Team:  <Team name> Team-Coordination association: <Coordination> → <Coordinated Team> <Coordinator Team> → <Coordination>
<i>Relation to other views/viewpoints</i>	- Team entity is common for all viewpoints. - Coordination extends Function defined in Process Viewpoint.

Table 5.4 Coordination Viewpoint

Suppose that there are two coordination cases in the project defined in the previous examples. In the first coordination case, Requirement Management Team coordinates Architecture Team. This coordination occurs over the requirements of the project. The second coordination occurs between Architecture and Development Teams. Architecture Team coordinates Development Teams through some checkpoints. These two coordination instances are shown in Figure 5.3 that shows the Coordination view of the example project case.

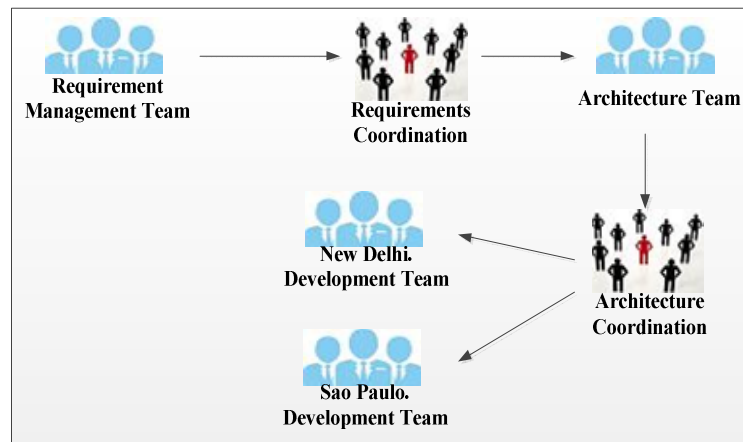


Figure 5.3 Coordination View: Example Case

5.5 Data Viewpoint

Ownership and physical deployment of data in GSD projects are represented in *Data Viewpoint*. This viewpoint's guide is shown in Table 5.5.

Viewpoint Element	Description
<i>Name</i>	Data Viewpoint
<i>Element Types</i>	Language, Team, Site, Data Storage, Digital Format, Actual Format, Data Entity
<i>Relation Types</i>	in, owned by, stored in, located by, can store
<i>Properties of Elements</i>	All elements: name, description, id Data Storage: canStoreDigital Data Entity: size, isDigital, sizeType, creationDate, lastUpdateDate
<i>Topology Constraints</i>	- Data Storage can store Digital Format only if Data Storage can store digital. - Data Entity is in some Digital Format only if Data Entity is digital.

Notation	Site:  <Site name> <Data Storages> Team:  <Team name> Data Storage:  <Data Storage name> <Compatible Formats> <Data Entities> Data Entity:  <Data Entity name> <Formats> Actual-Digital Format:  <Format name> Team-Data Storage association: <Data Storage> → <Team>
Relation to other views/viewpoints	- Team entity is common for all viewpoints. - Data Entity entity is used Process and Communication viewpoints. - Data Storage is used in Migration viewpoint. - Actual and Digital Formats are used in Tool Viewpoint.

Table 5.5 Data Viewpoint

Consider the example given in above sections. In the default way, each site has its own data storage and if some data is supposed to be shared, a local copy of the original data is created using a version control software like SVN. Suppose that the example follows the default way: New York, California, New Delhi and Sao Paulo sites have their local data storages and each site has its own documents and local copies of other necessary documents kept in these local storages. This situation is described in Figure 5.4.

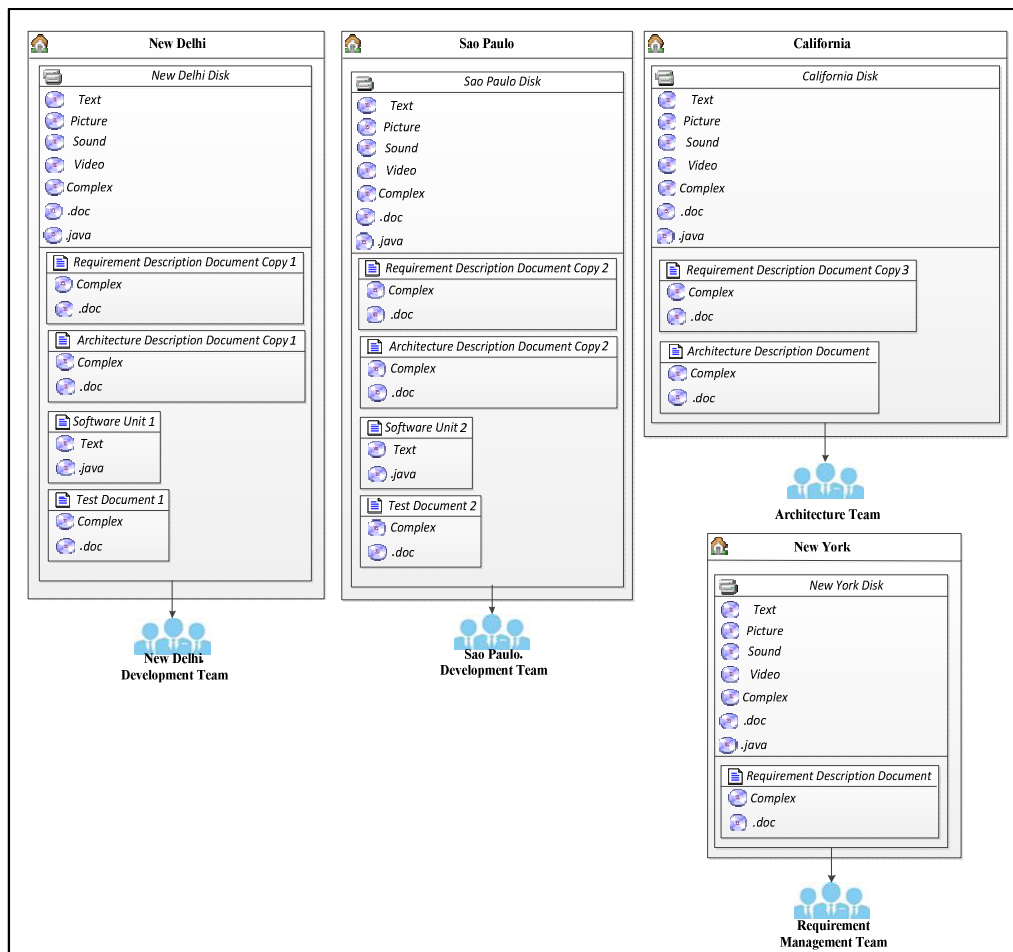


Figure 5.4 Data View: Example Case

5.6 Communication Viewpoint

Teams' communications during GSD project is captured in Communication Viewpoint. The guide of this viewpoint is in Table 5.6.

Viewpoint Element	Description
Name	Communication Viewpoint
Element Types	Team, Communication, Process, Data Entity, Communication Platform
Relation Types	has caller as, has receiver as, carries, aims, done over
Properties of Elements	All elements: name, description, id
Topology Constraints	- None.
Notation	

	Data Entity:  <Data Entity name> Team:  <Team name> Process:  <Process name> Communication:  <Communication name> <Data Entities> Communication Platform: <Communication Platform name> Team-Communication association: <Caller Team> → <Communication> <Communication> → <Receiver Team> Communication-Process association: <Communication> → <Process> Communication-Platform association: <Communication> → <Platform>
Relation to other views/viewpoints	- Team entity is common for all viewpoints. - Process entity is used in Process viewpoint. - Data Entity entity is used in Data viewpoint.

Table 5.6 Communication Viewpoint

Figure 5.5 shows an example view from the communication activities scenario of Requirement Analysis and Architecture Teams. After Requirement Team's Requirement Description Document production, Architecture Team receives the document and reviews it in order to create Architecture Description Document. During reviewing process, some questions are aroused and Architecture Team sends and gets reply of a bunch of emails to Requirement Team. When the textual communication is not enough to express ideas or feelings, Architecture Team makes phone calls, too.

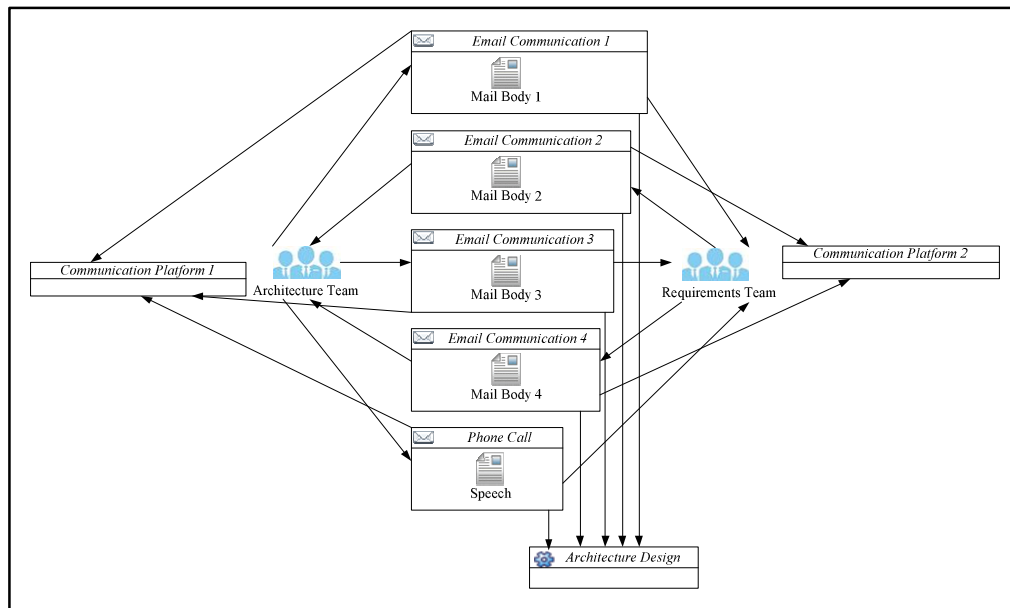


Figure 5.5 Communication View: Example Case

5.7 Tool Viewpoint

In order to provide some services and achieve communication, teams in GSD projects use tools. *Tool Viewpoint* aims to capture this tool usage. This viewpoint's guide is presented in Table 5.7.

Viewpoint Element	Description
<i>Name</i>	Tool Viewpoint
<i>Element Types</i>	Communication Platform, Function Platform, Team, Digital Format, Actual Format, Tool
<i>Relation Types</i>	used by, consists of, compatible with
<i>Properties of Elements</i>	All elements: name, description, id Tool: supportCollaboration
<i>Topology Constraints</i>	- None.

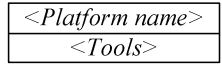
<i>Notation</i>	Tool:  <i><Tool name></i> <i><Compatible Formats></i> Team:  <i><Team name></i> Communication/Function Platform:  <i><Platform name></i> <i><Tools></i> Actual-Digital Format:  <i><Format name></i> Team-Platform association: <i><Platform></i> → <i><Team></i>
<i>Relation to other views/viewpoints</i>	- Team entity is common for all viewpoints. - Actual and Digital Formats are used in Data Viewpoint. - Communication Platform is used in Communication Viewpoint.

Table 5.7 Tool Viewpoint

As an example, we will provide the tool usage under the environment settings given in above examples. There are four sites of this example project, and each site has its own tools for serving particular purposes. Requirement Team uses Doors [3] in order to manage requirements, Architecture Team makes their designs in Enterprise Architect [4], Development Teams build their software using Eclipse Integrated Development Environment [5] and Test Teams also use Eclipse IDE, too.

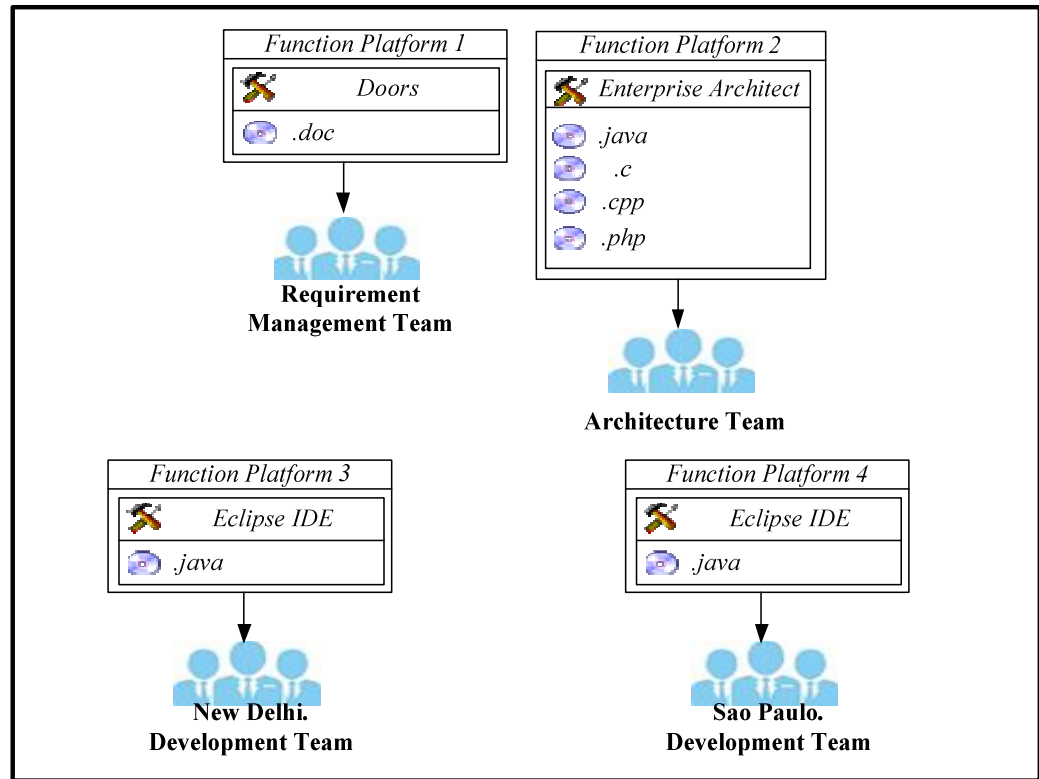


Figure 5.6 Tool View: Example Case

The figure above presents the tool usage of teams for providing main functionalities. One can use viewpoint definition in order to show communication tool usage, too. In this example, we keep the function platforms separate from the communication platforms to purify the ideas that we wish to express.

5.8 Migration Viewpoint

Migration Viewpoint has the functionality of representing traveling activities of GSD teams during project life cycle. Viewpoint guide of *Migration Viewpoint* is shown in Table 5.8.

Viewpoint Element	Description
Name	Migration Viewpoint
Element Types	Team, Process, Data Storage, Site, Migration

<i>Relation Types</i>	executed by, done in context of, done from/to, carries
<i>Properties of Elements</i>	All elements: name, description, id Migration: date
<i>Topology Constraints</i>	- Source and destination Sites for a Migration can't be the same.
<i>Notation</i>	Site:  <Site name> Team:  <Team name> Process:  <Process name> Migration:  <Migration name> <Processes> <Data Storages> <Teams> Data Storage:  <Data Storage name> Migration-Site association: <Home Site> → <Migration> <Migration> → <Destination Site>
<i>Relation to other views/viewpoints</i>	- Team entity is common for all viewpoints. - Process is used in Process viewpoint. - Data Storage is used in Data Storage viewpoint.

Table 5.8 Migration Viewpoint

As an example of Migration view, we will reconsider the case described in the previous sections. Suppose that for clarifying requirements and coordinating architecture design activities, Requirement Team processes a travel from New York, where they are located, to California to come together with Architecture Team. The travel is done in the context of Requirement Management and Architecture Design. Requirement Team also carries an external disk in order to carry some information that may be needed. This migration activity is shown in

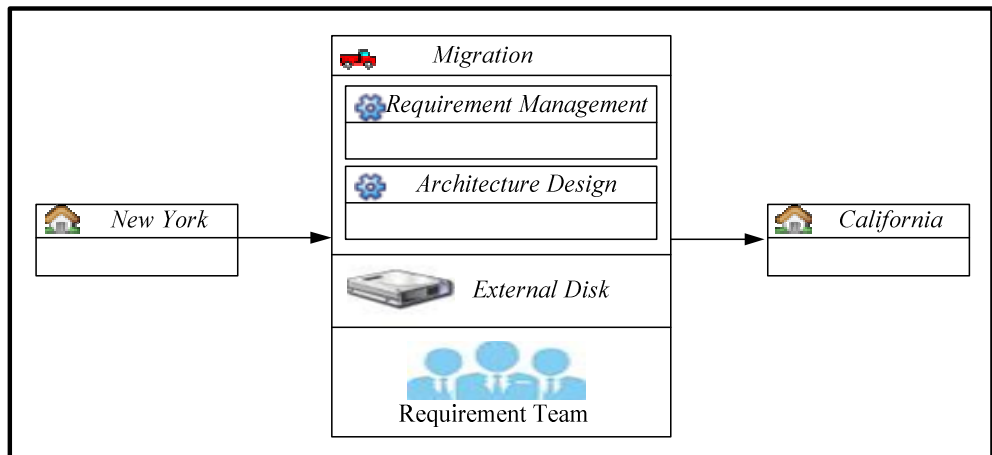


Figure 5.7 Migration View: Example Case

Chapter 6

Deriving Application Architecture Using Question Framework

The previous sections have defined a set of viewpoints that can be used to model GSD architectures. Designing the architecture for GSD systems is not trivial and requires in-depth knowledge of the GSD concerns. To support the GSD architect in designing a proper architecture, we propose a so-called question framework. The question framework includes a set of questions based on the previously defined meta-model, as well as the related answers and design actions that need to be taken to derive the architecture.

The chapter is organized as follows: Section 6.1 presents first the adopted approach for deriving GSD application architecture. In Section 6.2, the question framework is explained. In subsequent sections, the different question sets for the concerns Deployment, Process, Data, Communication, Tool and Migration are described.

6.1 Adopted Approach for Deriving Application Architecture

The adopted approach for deriving the application architecture for GSD is shown in Figure 6.1. The ordering of the steps are shown using numbers.

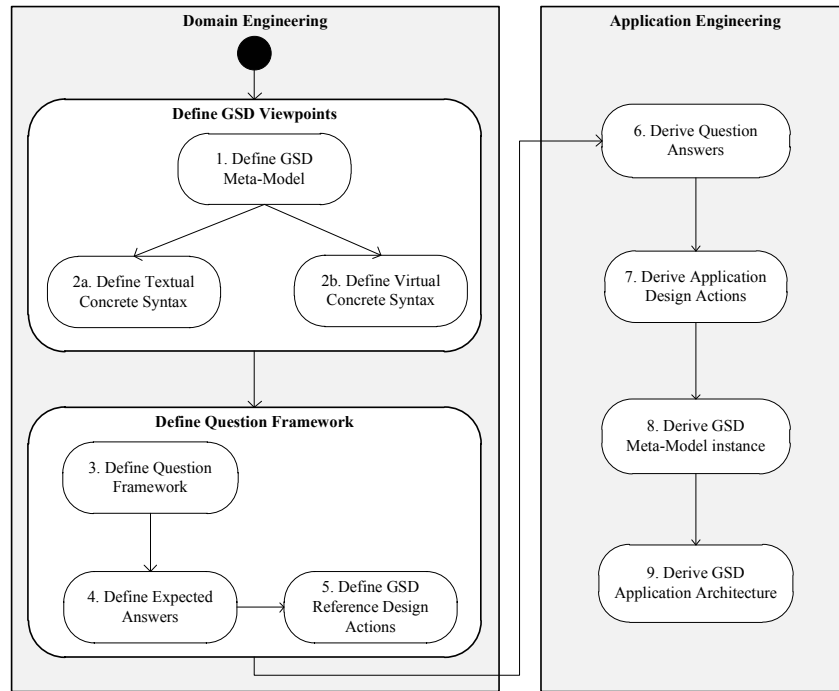


Figure 6.1 Adopted Approach for Deriving Application Architecture

The approach consists of two different types of activities including domain engineering and application engineering. In the domain engineering phase, the meta-model for GSD, and based on this the question framework is defined. The meta-model for GSD has been described in chapter 5, and based on this in chapter 6 the architectural viewpoints have been defined. In this chapter, we will elaborate on the question framework.

Once the meta-model, the architectural viewpoints and the related question framework have been defined the architect can start designing GSD. This is defined in the application engineering phase that provides answers to the questions in the question framework. Based on the provided answers, the related

design actions are derived. The design actions are further used to instantiate the application architecture based on the meta-model for GSD.

6.2 Overview of the Question Framework

The question framework is divided into six question sets that consist of a number of questions. Each set corresponds to one of the six meta-model units as described in the meta-model section. Like meta-model units, question sets are largely orthogonal but there are also intersecting concepts. The questions are numbered in a particular order to guide the proper derivation of the application architecture. The conceptual model for the question framework is given in Figure 6.2 **Question Framework Concepts Model**.

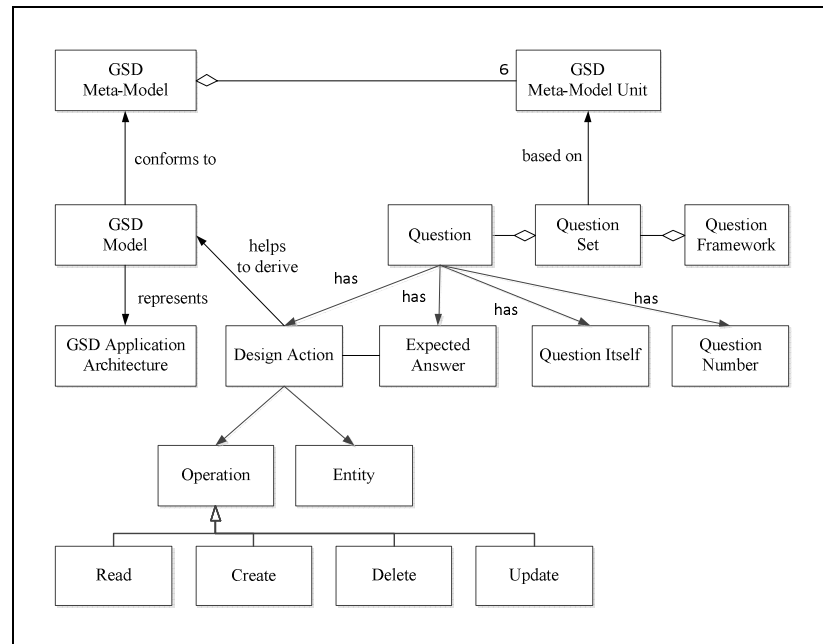


Figure 6.2 Question Framework Concepts Model

Each question is defined as a tuple consisting of the *index number* of the question, the *question itself*, the *expected answer* and the related *design action*. The *index number* is a unique number for identifying the particular question and for defining the order of the question. The question is described in natural

language. The expected answers are specific to the questions. It might include values of primitive data types (such as integer) or require more complicated answers. Related to the expected answers are the *design actions*. A design action defines a CRUD (create, read, update and delete) action to create, read, update or delete the elements of an entity in the model representing the application architecture. The actions on which these CRUD operations can apply are in principle *Entity*, *Entity Attribute* and *Association*. Based on this we have defined all the possible combinations for CRUD operations. For example, we have defined the following types of design actions:

Create Entity

Read Entity

Delete Entity

Update Entity

Create Association

Read Association

Delete Association

Update Association

Etc.

The design action is defined as a tuple consisting of <operation><entity >. The <operation> part represents one of the four CRUD operations. The part <entity> represents the elements in the model which corresponds to application architecture. Example design actions are *Create Team*, *Update Data Entity*, and *Create Expertise Area-Expertise Area Association*.

In addition to the regular CRUD-based design actions, we have also introduced so-called *Guide* design actions. *Guide* design actions are not related to entities but they are used to support the questions and other design actions. A guide design action is, for example, limiting the number of possible selections to a list.

In the following section we will explain each question set in more detail.

6.3 Deployment Question Set

Deployment Question Set includes 19 questions that are related to the Deployment meta-model unit. The question set can be found in Table 6.1. This question set includes questions to identify sites and teams, and their properties. Regarding the ordering of the questions first the details about sites and then the details about teams are asked.

All the questions except question numbered 5 are CRUD design actions. Question 5 determines the list of types of organizations such as commercial, open source community, etc.

No	Question	Expected Answer	Design Action
1	How many Sites are there in this GSD project?	Positive integer.	<Create Model> <Site>
2	Enter Site details.	Name, description and location are given in free text format. Time zone is selected from time zones in the world.	<Update Model-Attribute> <Site>.
3	Select parent Site for each Site.	A defined Site or null value is selected for each Site.	<Create Association> <Site-Site>
4	Which Languages are used?	Selections are done from languages spoken in the world.	<Create Model> <Language>
5	What are Organization types?	Free text. Ex: Comercial, Open Source Development, etc.	<Guide> Determine the list of types for Organizations.
6	How many Organizations are there in this GSD project?	Positive integer.	<Create Model> <Organization>
7	Enter Organization details.	Name and description are given in free text format. Type is chosen from the values given in Question 6's answer.	<Update Model-Attribute> <Organization>
8	Define Social Cultures.	Name and description for each instance that is wanted to be created are given in free text format.	<Create Model> <Social Culture>
9	Define Work Cultures.	Name and description for each instance that is wanted to be created are given in free text format.	<Create Model> <Work-Culture>

10	How many Expertise Areas are there in this GSD project?	Positive integer.	<Create Model> <Expertise Area>
11	Enter Expertise Area details.	Name and description are given in free text format.	<Update Model-Attribute> < Expertise Area>
12	Select parent Expertise Area for each Expertise Area.	A defined Expertise Area or null value is selected as parent for each Expertise Area.	<Create Association> <Expertise Area Expertise Area>
13	How many Teams are there in this GSD project?	Positive integer.	<Create Model> <Team>
14	Enter Team details.	Name and description are given in free text format. isTemporary and isVirtual attributes are given in boolean values.	<Update Model-Attribute> <Team>
15	Select Site and Organization for each Team.	A defined Organization and Site or null value is selected for each Team instance.	<Create Association> <Team-Site> <Team-Organization>
16	Select Work and Social Cultures for each Team.	At least one defined Work Culture and Social Culture are selected for each Team.	<Create Association> < Team-Work Culture>; <Create Association> < Team-Social Culture>
17	Select Languages that each Team able to speak.	At least one defined Language defined before is selected for each Team.	<Create Association> <Team-Language>
18	Select Expertise Areas for each Team.	At least one defined Expertise Area is selected for each Team.	<Create Association> <Team-Expertise>
19	Select parent Team for each Team.	A defined Team or null value is selected as parent for each Team.	<Create Association> <Team-Team>

Table 6.1 Deployment Question Set

6.4 Process Question Set

Process Question Set has 21 questions associated with it. The question set includes questions related to both Process viewpoint and Coordination viewpoint. The question set is given in Table 6.2. This question set aims to describe the processes and the entities associated with these processes. Entities are, for example, participant teams, input and output data entities, or functions that each process provides.

No	Question	Expected Answer	Design Action
1	How many Expertise Areas are there in this GSD project?	Positive integer.	<Create Model> <Expertise Area>
2	Enter Expertise Area details.	Name and description are given in free text format.	<Update Model-Attribute> < Expertise Area>
3	Select parent Expertise Area for each Expertise Area.	A defined Expertise Area or null value is selected as parent for each Expertise Area.	<Create Association> <Expertise Area Expertise Area>
4	How many Teams are there in this GSD project?	Positive integer.	<Create Model> <Team>
5	Enter Team details.	Name and description are given in free text format. isTemporary and isVirtual attributes are given in boolean values.	<Update Model-Attribute> <Team>
6	Select Expertise Areas for each Team.	At least one defined Expertise Area is selected for each Team.	<Create Association> <Team-Expertise>
7	Select parent Team for each Team.	A defined Team or null value is selected as parent for each Team.	<Create Association> <Team-Team>
8	How many Functions are there in this GSD project?	Positive integer.	<Create Model> <Function>
9	Enter Function details.	Name and description are given in free text format. Coordination instances also distinguished by a boolean question.	<Update Model-Attribute> <Function>
10	Select Expertise Areas that are needed for serving each Function.	At least one Expertise Area instance defined before is selected for each Function instance.	< Create Association> < Function-Expertise Area>
11	Select parent Function for each Function.	A Function instance defined before or null value is selected as parent for each Function instance.	< Create Association> < Function-Function >
12	Select Teams who will be coordinated in each Coordination instance.	At least two Team instances defined before are selected for each Coordination instance.	< Create Association > <Coordination-Team>
13	Define Processes.	Name and description are in free text format.	<Create Model> <Process>
14	Select Teams who participate in each Process.	At least one Team instance defined before is selected for each Process instance.	<Create Association> <Process-Team>
15	Select Functions that will be served by each Process.	At least one Function instance defined before is selected for each Process instance.	<Create Association> <Process-Function>
16	What are size types for Data Entities?	Free text. Ex: page, gb, minutes, pixels, etc.	<Guide> Determine the list of size types for Data Entities instances.
17	How many Data Entities are there in this GSD project?	Positive integer.	<Create Model> <Data Entity>
18	Enter Data Entity details.	Name and description are in free text format. isDigital attribute is	<Update Model-Attribute>

		in boolean value. Size attribute is in double value. SizeType is chosen from the values given in the Question 17's answer. CreationDate and lastUpdateDate are in date format.	<Data Entity>
19	Select Process for each Data Entity.	A defined Process is selected for each Data Entity.	<Create Association> <Data Entity-Process>
20	Select parent Data Entity for each Data Entity.	A Data Entity instance defined before or null value is selected as parent for each Data Entity instance.	<Create Association> <Data Entity-Data Entity>
21	Select Data Entities used by each Process.	Zero or more Data Entities defined before are selected for each Process instance.	<Create Association> <Process-Data Entity>

Table 6.2 Process Question Set

6.5 Data Question Set

Data Question Set includes 21 questions related to Data meta-model. The question set can be found in Table 6.3. This question set is represented for capturing the data entities existing in a GSD project. Except question 10, all questions are associated with CRUD design actions. Question 10 asks for the size types of data entities.

While defining the question set, we assumed that five pre-defined Actual Format instances exist: video, sound, text, picture and complex. Complex instance is used when a Data Entity is in hierarchy, in other words, it is included in some other Data Entity as child.

No	Question	Expected Answer	Design Action
1	How many Teams are there in this GSD project?	Positive integer.	<Create Model> <Team>
2	Enter Team details.	Name and description are given in free text format. isTemporary and isVirtual attributes are given in boolean values.	<Update Model-Attribute> <Team>
3	Select parent Team for each Team.	A defined Team or null value is selected as parent for each Team.	<Create Association> <Team-Team>
4	Which Languages are used?	Selections are done from languages spoken in the world.	<Create Model> <Language>

5	How many Sites are there in this GSD project?	Positive integer.	<Create Model> <Site>
6	Enter Site details.	Name, description and location are given in free text format. Time zone is selected from time zones in the world.	<Update Model-Attribute> <Site>.
7	Select parent Site for each Site.	A defined Site or null value is selected for each Site.	<Create Association> <Site Site>
8	Define Digital Formats.	Name and description for each Digital Format instance that is wanted to created are in free text format.	<Create Model> <Digital Format>
9	Define Actual Formats.	Name and description for each Actual Format instance that is wanted to created are in free text format.	<Create Model> <Actual Format>
10	What are size types for Data Entities?	Free text. Ex: page, gb, minutes, pixels, etc.	<Guide> Determine the list of size types for Data Entities instances.
11	How many Data Entities are there in this GSD project?	Positive integer.	<Create Model> <Data Entity>
12	Enter Data Entity details.	Name and description are in free text format. isDigital attribute is in boolean value. Size attribute is in double value. SizeType is chosen from the values given in the Question 17's answer. CreationDate and lastUpdateDate are in date format.	<Update Model-Attribute> <Data Entity>
13	Select parent Data Entity for each Data Entity.	A Data Entity instance defined before or null value is selected as parent for each Data Entity instance.	<Create Association> <Data Entity-Data Entity>
14	Select Actual Format and (if digital) Digital Format of each Data Entity.	An Actual Format instance and a Digital Format instance(if Data Entity is digital) defined before are selected for each Data Entity instance.	<Create Association> <Data Entity-Actual Format> <Data Entity-Digital Format>
15	How many Data Storages are there in this GSD project?	Positive integer.	<Create Model> <Data Storage>
16	Enter Data Storage details.	Name and description are in free text format. canStoreDigital attribute is in boolean value.	<Update Model-Attribute> <Data Storage>
17	Select Actual Formats and (if digital) Digital Formats that each Data Storage can store.	At least one Actual Format and Digital Format(if it can store digital) instances defined before are selected for each Data Storage instance.	<Create Association> <Data Storage-Actual Format> <Data Storage-Digital Format>
18	Select Languages for each Data Entity.	At least Language instance defined before are selected for each Data Entity.	<Create Association> <Data Entity-Language>
19	Select parent Data Storage for each Storage.	A Data Storage instance defined before or null value is selected as	<Create Association> <Data Storage-Data

		parent for each Data Storage instance.	Storage>
20	Select owner Team and Site for each Data Storage.	Teams and Sites defined before are selected for each Data Storage instances.	<Create Association> <Data Storage-Team> <Data Storage-Site>
21	Select Data Entities that each Data Storage stores.	Data Entities defined before are selected for each Data Storage instances.	<Create Association> <Data Storage-Data Entity>

Table 6.3 Data Question Set

6.6 Communication Question Set

Communication Question Set has 15 questions. The question set is given in Table 6.4. This question set includes questions related to Communication viewpoint. It aims to identify the communication activities occurring in GSD projects with the associated entities such as communicating teams or context of the communication instance.

In question 13, at least one caller team must be selected and no team as receiver may be selected. This depends on the communication type decided or each Communication instance. Communication types are defined by question 10 which is associated with a guide design action.

Communication-Communication Platform association is not included in this question set since it is already included in Communication Question Set. In this question set, we included Function-Function Platform association.

No	Question	Expected Answer	Design Action
1	How many Teams are there in this GSD project?	Positive integer.	<Create Model> <Team>
2	Enter Team details.	Name and description are given in free text format. isTemporary and isVirtual attributes are given in boolean values.	<Update Model-Attribute> <Team>
3	Select parent Team for each Team.	A defined Team or null value is selected as parent for each Team.	<Create Association> <Team-Team>
4	What are size types for Data Entities?	Free text. Ex: page, gb, minutes, pixels, etc.	<Guide> Determine the list of size

			types for Data Entities instances.
5	How many Data Entities are there in this GSD project?	Positive integer.	<Create Model> <Data Entity>
6	Enter Data Entity details.	Name and description are in free text format. isDigital attribute is in boolean value. Size attribute is in double value. SizeType is chosen from the values given in the Question 17's answer. CreationDate and lastUpdateDate are in date format.	<Update Model-Attribute> <Data Entity>
7	Select parent Data Entity for each Data Entity.	A Data Entity instance defined before or null value is selected as parent for each Data Entity instance.	<Create Association> <Data Entity-Data Entity>
8	Define Processes.	Name and description are in free text format.	<Create Model> <Process>
9	Define Communication Platforms.	Name and description are in free text format.	<Create Model> <Communication Platform>
10	What are types for Communication?	Free text. Email, telephone, fax, etc.	<Guide> Determine the list of types for Communication instances.
11	Define Communications.	Name and description are in free text format. isFormal, isSynchronous and isChannel attributes are in boolean value. Type is chosen from the values given in Question 10's answer. If Communication instance is not a channel, then startTime and finishTime are given in date format. suggestedTimePeriod attribute is selected as two time values.	<Create Model> <Communication>
12	Select Communication Platform for each Communication.	A Communication Platform instance defined before is selected for each Communication instance.	<Create Association> <Communication-Communication Platform>
13	Select caller and receiver Teams for each Communication.	At least one caller Team and receiver Teams defined before are selected for each Communication instance.	<Create Association> <Communication-Team>
14	Select Data Entities carried by each Communication.	At least one Data Entity instance defined before is selected for each Communication instance.	<Create Association> <Communication-Data Entity>
15	Select Processes aimed by each Communication.	At least one Process instance is selected for each Communication instance.	<Create Association> <Communication-Process>

Table 6.4 Communication Question Set

6.7 Tool Question Set

Tool Question Set includes 17 questions related to Tool meta-model unit. There is no question including a Guide Design Action. Tool Question Set is given in Table 6.5. The set is used to clarify the tools used in a GSD project for communication and service purposes.

No	Question	Expected Answer	Design Action
1	How many Teams are there in this GSD project?	Positive integer.	<Create Model> <Team>
2	Enter Team details.	Name and description are given in free text format. isTemporary and isVirtual attributes are given in boolean values.	<Update Model-Attribute> <Team>
3	Select parent Team for each Team.	A defined Team or null value is selected as parent for each Team.	<Create Association> <Team-Team>
4	Define Digital Formats.	Name and description for each Digital Format instance that is wanted to created are in free text format.	<Create Model> <Digital Format>
5	Define Actual Formats.	Name and description for each Actual Format instance that is wanted to created are in free text format.	<Create Model> <Actual Format>
6	Define Function Platforms.	Name and description are in free text format.	<Create Model> <Function Platform>
7	Define Communication Platforms.	Name and description are in free text format.	<Create Model> <Communication Platform>
8	How many Functions are there in this GSD project?	Positive integer.	<Create Model> <Function>
9	Enter Function details.	Name and description are given in free text format. Coordination instances also distinguished by a boolean question.	<Update Model-Attribute> <Function>
10	Select parent Function for each Function.	A Function instance defined before or null value is selected as parent for each Function instance.	< Create Association> < Function-Function >
11	Select Function Platform for each Function.	At least one Function Platform instance defined before is selected for each Function instance.	<Create Association> <Function-Function Platform>
12	Define Tools.	Name and description are in free text format. SupportCollaboration attribute is in boolean value.	<Create Model> <Tool>
13	Select compatible Actual and Digital Formats with each	At least one Actual Format and Digital Format instances defined	<Create Association> <Tool-Actual Format>

	Tool.	before are selected for each Tool instance.	<Tool-Digital Format>
14	Select Tools for each Function Platform.	At least one Tool instance defined before is selected for each Function Platform instance.	<Create Association> <Function Platform-Tool>
15	Select Tools for each Communication Platform.	At least one Tool instance defined before is selected for each Communication Platform instance.	<Create Association> <Communication Platform-Tool>
16	Select owner Team of each Function Platform.	At least one Team instance defined before is selected for each Function Platform instance.	<Create Association> <Function Platform-Team>
17	Select owner Team of each Communication Platform.	At least one Team instance defined before is selected for each Communication Platform instance.	<Create Association> <Communication Platform-Team>

Table 6.5 Tool Question Set

6.8 Migration Question Set

In *Migration Question Set*, there are 15 questions. This question set has questions related to traveling activities of GSD teams during GSD projects. Migration Question Set can be found in Table 6.6.

Considering the order of the questions, before the definition of Migration instances, the Team, Site, Process and Data Storage instances should be defined in order to create Migration-* associations.

No	Question	Expected Answer	Design Action
1	How many Teams are there in this GSD project?	Positive integer.	<Create Model> <Team>
2	Enter Team details.	Name and description are given in free text format. isTemporary and isVirtual attributes are given in boolean values.	<Update Model-Attribute> <Team>
3	Select parent Team for each Team.	A defined Team or null value is selected as parent for each Team.	<Create Association> <Team-Team>
4	How many Data Storages are there in this GSD project?	Positive integer.	<Create Model> <Data Storage>
5	Enter Data Storage details.	Name and description are in free text format. canStoreDigital attribute is in boolean value.	<Update Model-Attribute> <Data Storage>
6	Select parent Data Storage	A Data Storage instance defined	<Create Association>

	for each Storage.	before or null value is selected as parent for each Data Storage instance.	<Data Storage-Data Storage>
7	Define Processes.	Name and description are in free text format.	<Create Model> <Process>
8	How many Sites are there in this GSD project?	Positive integer.	<Create Model> <Site>
9	Enter Site details.	Name, description and location are given in free text format. Time zone is selected from time zones in the world.	<Update Model-Attribute> <Site>.
10	Select parent Site for each Site.	A defined Site or null value is selected for each Site.	<Create Association> <Site Site>
11	Define Migrations.	Name and description are in free text format. Date attribute is in date value.	<Create Model> <Create Migration>
12	Select Teams who migrates.	At least one Team instance defined before is selected for each Migration instance.	<Create Association> <Migration-Team>
13	Select home and destination Sites for each Migration.	Two Site instances defined before are selected one as from and the second one as to location for each Migration instance.	<Create Association> <Migration-Site>
14	Select Data Storages carried by each Migration.	Data Storage instances defined before are selected for each Migration instance.	<Create Association> <Migration-Data Storage>
15	Select Processes that each Migration is done in the context of.	Process instances defined before are selected for each Migration instance.	<Create Association> <Migration-Process>

Table 6.6 Migration Question Set

Chapter 7

Tool Support for Generating Application Architecture

In this chapter, we discuss the tool support that is needed for modeling and deriving GSD architecture. Section 7.1 provides a general overview of the tool support. In section 7.2, we present *Global Architect* that we have developed for realizing the approach. Section 7.3 discusses the domain-specific languages that we have developed using XText. Finally, in Section 7.4, we conclude with the Graphical Modeling Framework to derive the visual notations for the defined views.

7.1 Overview of Tool Support

In principle, it is possible to create an application architecture by using the meta-model and the question framework manually. However, answering all the questions and keeping track of the required design actions can be a cumbersome and error-prone activity. Actually, since the relation among question, expected answers and design actions are now formally defined, we have provided automated support for the approach, instead. The automation workflow is displayed in Figure 7.1.

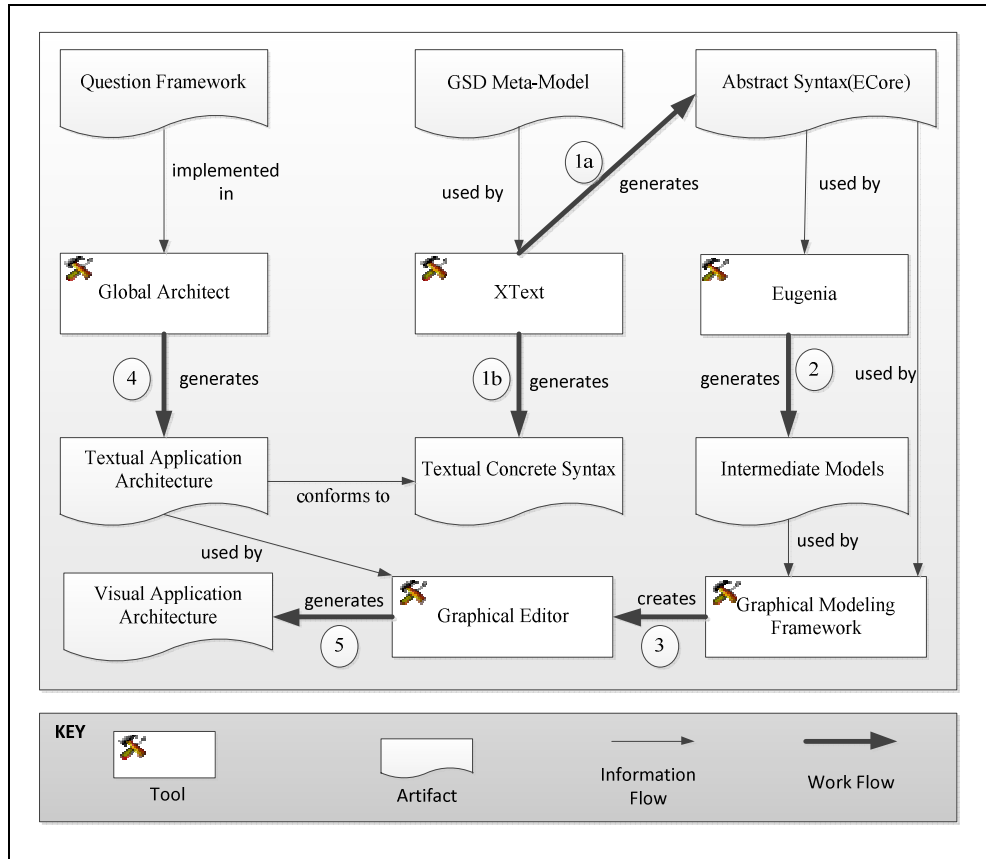


Figure 7.1 Tool Support Activities for Deriving Application Architecture

Using the XText [6] tool the syntax of the GSD meta-model (1b) is generated. During this process, XText also produces abstract syntax as an ECore model (1a). Then, with the help of Eugenia [9], the ECore model generated by XText is used in the generation of intermediate models (2). After that, the intermediate models and the ECore model are merged together using the Graphical Modeling Framework [7] in order to create visual editor for the viewpoint (3). The question framework is implemented in the tool so-called *Global Architect*. The architect answers questions through Global Architect and the tool creates output of application architecture in textual form which conforms to desired viewpoint's language defined using XText [6] tool (4). Finally, the textual application architecture is converted to the visual form (5).

7.2 Global Architect

Global Architect is a Java application which implements GSD meta-model and the question framework with the purpose of helping the architect by automating the application architecture derivation process. A screenshot from Global Architect can be found in Figure 7.2.

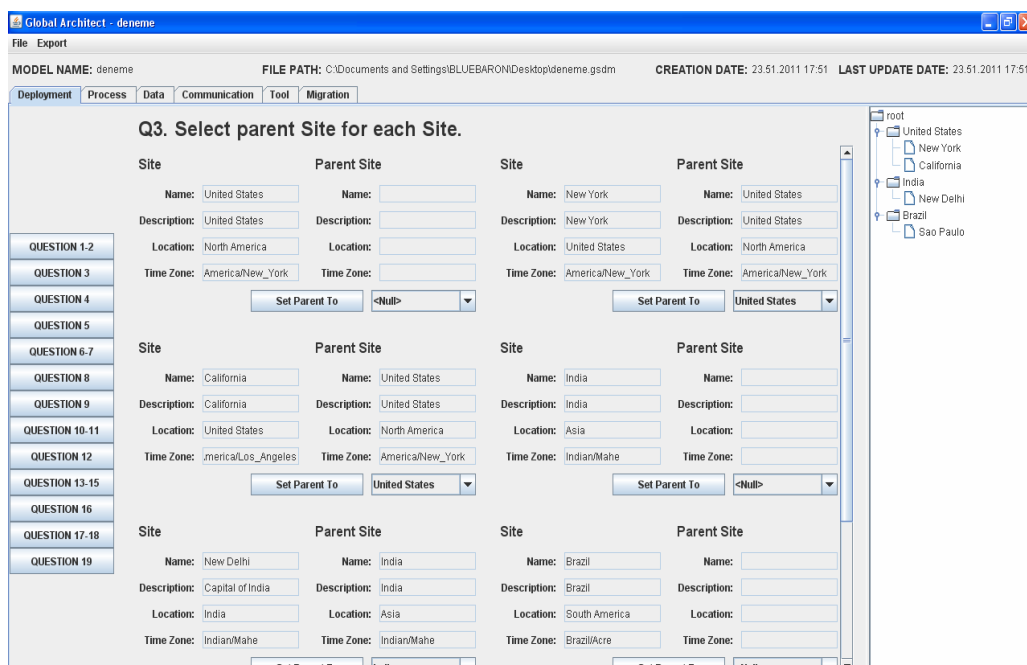


Figure 7.2 Screenshot from Global Architect

The tool is a Java application consisting of 3 layers. General user interface components and interfaces of each question are located in the first layer. The second layer is application logic. Application logic layer provides communication function between interface and the model layers. The last layer, which is model layer, consists of meta-model classes corresponding to the representation of the meta-model entities defined before.

Global Architect has one main screen shown in Figure 7.2. The menu bar at the top has two menus. The file menu provides basic functionalities such as creating a new model, saving and loading models. The model can be exported by options under export menu.

After a model is created, information about the model is shown in the top panel. The model name, file path, creation and last update dates are displayed here. The model name is the same as the file name. Models are saved with *.gsdm (Global Software Development Model) extension.

The main screen's body has six tabs each corresponds to one of the question sets in the question framework. The questions related to each question set are shown on the left panel. The architect needs to answer these questions to get the related view of application architecture. The architect can navigate through each of these questions. Some questions are grouped together for showing a nice presentation. The tool itself defines the required ordering to answer questions. The selected question is shown in the middle panel.

The answers to the questions result in updates of the model, which is an instance of the meta-model that we have defined before. The view on the partial model is shown on the right panel. In the given screenshot, the question relates to the structure of the sites in GSD (The screenshot shows the example given in Deployment Viewpoint section) and likewise the Site hierarchy part of the model is shown in the right panel.

Once the architect is ready with answering all the questions, the resulted model can be export in different formats by using the export menu. The model can be exported in seven different textual formats each corresponds to a viewpoint definition that we have given before. The format names start with gs meaning Global Software and end with ml meaning modeling language. The format names are as follows: *.gsdml (Deployment), *.gspml (Process),

*.gscoml (Coordination), *.gsdaml (Data), *.gscml (Communication), *.gstml (Tool) and *.gsmml (Migration). The textual outputs conform to the textual concrete syntax definitions of each viewpoint. The details of textual concrete syntaxes can be found in the following section.

7.3 XText

A domain-specific language (DSL) is *a small, usually declarative, language that offers expressive power focused on a particular problem domain* [10]. XText [6] is a language development framework as an Eclipse plug-in that allows users to design their own DSLs. It gives a full implementation of the language running on the Java virtual machine. The full implementation has compiler components such as parser, type-safe abstract syntax tree, serializer and code formatter. XText's runtime components interoperable with Eclipse Modeling Framework (EMF) [8] since they are based on EMF, which is a property of XText that we are going to use in order to satisfy the integration of our language with Graphical Modeling Framework.

We have defined seven domain-specific languages each of which corresponds to one of the GSD viewpoints that we have defined. These languages can be considered as textual concrete syntaxes of viewpoints. Each language is defined in a *.xtext file which is the default file extension of XText defining the language and in a form similar to EBNF and the language itself is in a similar form with XML, i.e. it has opening and closing tags. The language names start with GS meaning Global Software and end with ML meaning Modeling Language. The language names are the same with viewpoint names. The language instance files' extensions are given in the previous section as export options of the model in Global Architect.

In the following subsections, we have described each of the languages we defined for each viewpoint and gave an example of that language instance which corresponds to the example case given in the viewpoints chapter.

7.3.1 Global Software Deployment Modeling Language (GSDML)

Global Software Deployment Modeling Language is the textual concrete syntax of the Deployment viewpoint. The language is given in Figure 7.3.

```

1 grammar org.xtext.example.mydsl.GSDML with org.eclipse.xtext.common.Terminals
2
3 generate gsdml "http://www.xtext.org/example/mydsl/GSDML"
4
5 GSDModel:
6     '<Model' 'id=' id=INT 'name=' name=STRING 'creationDate=' creationDate=STRING
7         'lastUpdateDate=' lastUpdateDate=STRING '>'
8         '<Sites>'
9         (sites+=Site)+
10        '</Sites>'
11        '</Model>';
12
13 Language:
14     '<Language' 'id=' id=INT 'name=' name=ID '>';
15
16 Organization:
17     '<Organization' 'id=' id=INT 'name=' name=STRING 'description=' description=STRING
18         'type=' type=STRING '>';
19
20 SocialCulture:
21     '<SocialCulture' 'id=' id=INT 'name=' name=STRING 'description=' description=STRING '>';
22
23 WorkCulture:
24     '<WorkCulture' 'id=' id=INT 'name=' name=STRING 'description=' description=STRING '>';
25
26 ExpertiseArea:
27     '<ExpertiseArea' 'id=' id=INT 'name=' name=STRING 'description=' description=STRING '>';
28
29 Site:
30     '<Site' 'id=' id=INT 'name=' name=STRING 'description=' description=STRING
31         'timezone=' timezone=STRING 'location=' location=STRING
32         'childrenList=[' (childrenList+=Site) * ']'
33         'teamList=[' (teamList+=Team) * ']'
34     '>';
35
36 Team:
37     '<Team' 'id=' id=INT 'name=' name=STRING 'description=' description=STRING
38         'isVirtual=' isVirtual?='virtual'? 'isTemporary=' isTemporary?='temporary'?
39         'languageList=[' (languageList+=Language) + ']'
40         'organization=' organization=Organization?
41         'socialCultureList=[' (socialCultureList+=SocialCulture) + ']'
42         'workCultureList=[' (workCultureList+=WorkCulture) + ']'
43         'expertiseAreaList=[' (expertiseAreaList+=ExpertiseArea) + ']'
44         ('parent=' parent=[Team|STRING])?
45     '>';

```

Figure 7.3 Global Software Deployment Modeling Language

As you can see, the elements Language, Organization, Social Culture, Work Culture, Expertise Area, Site and Team in the meta-model are mapped to rules. At the top, the elements are unified in a model using the GSDModel rule.

Global Architect's output, which is an instance of this language, is given in Figure 7.4. The instance describes the example case explained in Deployment viewpoint section.

```

1 <Model id=68 name="deneme" creationDate="23.51.2011 17:51" lastUpdateDate="24.51.2011 11:43">
2   <Sites>
3     <Site id=74 name="United States" description="United States" timezone="America/New_York" location="North America"
4       childrenList=[
5         <Site id=75 name="New York" description="New York" timezone="America/New_York" location="United States"
6           childrenList=[ ]
7           teamList=[
8             <Team id=101 name="Requirement Management Team" description="Requirement Management Team" isVirtual= isTemporary=
9               languageList=[ <Language id=81 name="English/"> <Language id=82 name="Spanish/"> ]
10              organization=<Organization id=85 name="Company A" description="Company A" type="Commercial" />
11              socialCultureList=[ <SocialCulture id=87 name="American Culture" description="American Culture"/>]
12              workCultureList=[ <WorkCulture id=90 name="Work Culture 1" description="Work Culture 1"/>]
13              expertiseAreaList=[ <ExpertiseArea id=96 name="Requirement Analysis" description="Requirement Analysis"/> ]
14            ]
15          ]
16        ]
17      ]
18      <Site id=76 name="California" description="California" timezone="America/Los_Angeles" location="United States"
19        childrenList=[ ]
20        teamList=[
21          <Team id=102 name="Architecture Team" description="Architecture Team" isVirtual= isTemporary=
22            languageList=[ <Language id=81 name="English/"> <Language id=82 name="Spanish/">]
23            organization=<Organization id=85 name="Company A" description="Company A" type="Commercial" />
24            socialCultureList=[ <SocialCulture id=87 name="American Culture" description="American Culture"/>]
25            workCultureList=[ <WorkCulture id=91 name="Work Culture 2" description="Work Culture 2"/>]
26            expertiseAreaList=[ <ExpertiseArea id=97 name="Architecture Design" description="Architecture Design"/>]
27          ]
28        ]
29      ]
30    ]
31  ]
32  <Site id=77 name="India" description="India" timezone="Indian/Mahe" location="Asia"
33    childrenList=[
34      <Site id=78 name="New Delhi" description="New Delhi" timezone="Indian/Mahe" location="India"
35        childrenList=[ ]
36        teamList=[
37          <Team id=103 name="Development Team A" description="Development Team A" isVirtual= isTemporary=
38            languageList=[ <Language id=81 name="English/"> <Language id=84 name="Hindi/">]
39            organization=<Organization id=85 name="Company A" description="Company A" type="Commercial" />
40            socialCultureList=[ <SocialCulture id=88 name="Indian Culture" description="Indian Culture"/>]
41            workCultureList=[ <WorkCulture id=92 name="Work Culture 3" description="Work Culture 3"/>]
42            expertiseAreaList=[ <ExpertiseArea id=98 name="Java Development" description="Java Development"/>]
43          ]
44          <Team id=104 name="Test Team A" description="Test Team A" isVirtual= isTemporary=
45            languageList=[ <Language id=81 name="English/"> <Language id=84 name="Hindi/">]
46            organization=<Organization id=85 name="Company A" description="Company A" type="Commercial" />
47            socialCultureList=[ <SocialCulture id=88 name="Indian Culture" description="Indian Culture"/>]
48            workCultureList=[ <WorkCulture id=93 name="Work Culture 4" description="Work Culture 4"/>]
49            expertiseAreaList=[ <ExpertiseArea id=99 name="Software Testing" description="Software Testing"/>]
50          ]
51        ]
52      ]
53    ]
54  ]
55  <Site id=79 name="Brazil" description="Brazil" timezone="Brazil/Acre" location="South America"
56    childrenList=[
57      <Site id=80 name="Sao Paulo" description="Sao Paulo" timezone="Brazil/Acre" location="Brazil"
58        childrenList=[ ]
59        teamList=[
60          <Team id=105 name="Development Team B" description="Development Team B" isVirtual= isTemporary=
61            languageList=[ <Language id=81 name="English/"> <Language id=83 name="Portuguese/">]
62            organization=<Organization id=86 name="Company B" description="Company B" type="Commercial" />
63            socialCultureList=[ <SocialCulture id=89 name="Brazilian Culture" description="Brazilian Culture"/>]
64            workCultureList=[ <WorkCulture id=94 name="Work Culture 5" description="Work Culture 5"/>]
65            expertiseAreaList=[ <ExpertiseArea id=100 name="Enterprise Applications" description="Enterprise Applications"/>]
66          ]
67          <Team id=106 name="Test Team B" description="Test Team B" isVirtual= isTemporary=
68            languageList=[ <Language id=81 name="English/"> <Language id=83 name="Portuguese/">]
69            socialCultureList=[ <SocialCulture id=89 name="Brazilian Culture" description="Brazilian Culture"/>]
70            workCultureList=[ <WorkCulture id=95 name="Work Culture 6" description="Work Culture 6"/>]
71            expertiseAreaList=[ <ExpertiseArea id=99 name="Software Testing" description="Software Testing"/>]
72          ]
73        ]
74      ]
75    ]
76  ]
77  ]
78  </Sites>
79 </Model>

```

Figure 7.4 GSDML: Example Case

7.3.2 Global Software Process Modeling Language (GSPML)

Global Software Process Modeling Language represents the textual notation of the Deployment viewpoint. The language is given in Figure 7.5.

```
1 grammar org.xtext.example.mydsl.GSPML with org.eclipse.xtext.common.Terminals
2
3 generate gSPML "http://www.xtext.org/example/mydsl/GSPML"
4
5 GSPModel:
6     '<Model' 'id=' id=INT 'name=' name=STRING 'creationDate=' creationDate=STRING
7         'lastUpdateDate=' lastUpdateDate=STRING '>'
8         '<DataEntities>'
9         (dataEntities+=DataEntity) *
10        '</DataEntities>'
11        '<Functions>'
12        (functions+=Function) *
13        '</Functions>'
14        '<Processes>'
15        (processes+=Process) *
16        '</Processes>'
17    '</Model>';
18
19 Process:
20     '<Process' 'id=' id=INT 'name=' name=STRING 'description=' description=STRING
21     'teamList=[' (teamList+=Team) + ']'
22     'inputEntityList=[' (nEntityList+=[DataEntity|STRING]) * ']'
23     'outputEntityList=[' (oEntityList+=[DataEntity|STRING]) * ']'
24     'functionList=[' (functionList+=[Function|STRING]) * ']'
25     '>';
26
27 Team:
28     '<Team' 'id=' id=INT 'name=' name=STRING 'description=' description=STRING '>';
29
30 ExpertiseArea:
31     '<ExpertiseArea' 'id=' id=INT 'name=' name=STRING 'description=' description=STRING
32     ('parent=' parent=[ExpertiseArea|STRING])? '>';
33
34 DataEntity:
35     '<DataEntity' 'id=' id=INT 'name=' name=STRING 'description=' description=STRING
36     ('parent=' parent=[DataEntity|STRING])? '>';
37
38 Function:
39     '<Function' 'id=' id=INT 'name=' name=STRING 'description=' description=STRING
40     'expertiseAreaList=[' (expertiseAreas+=ExpertiseArea) * ']'
41     ('parent=' parent=[Function|STRING])? '>';
```

Figure 7.5 Global Software Process Modeling Language

GSPModel rule, the starting rule, has Data Entity, Function and Process lists. The other entities included in Process meta-model are connected to these entities by containment relationship.

For the example case described in Process viewpoint section, Global Architect creates the *.gspml output given in Figure 7.6.

```

1 <Model id=68 name="deneme" creationDate="23.51.2011 17:51" lastUpdateDate="24.51.2011 12:37">
2   <DataEntities>
3     <DataEntity id=114 name="Requirement Description Document" description="Requirement Description Document" />
4     <DataEntity id=115 name="Architecture Description Document" description="Architecture Description Document" />
5     <DataEntity id=116 name="Software Unit 1" description="Software Unit 1" />
6     <DataEntity id=117 name="Software Unit 2" description="Software Unit 2" />
7   </DataEntities>
8   <Functions>
9     <Function id=107 name="Requirement Derivation" description="Requirement Derivation"
10      expertiseAreaList=[<ExpertiseArea id=96 name="Requirement Analysis" description="Requirement Analysis"/> ]/>
11     <Function id=108 name="Architecture Design" description="Architecture Design"
12      expertiseAreaList=[<ExpertiseArea id=97 name="Architecture Design" description="Architecture Design"/> ]/>
13     <Function id=109 name="Java Programming" description="Java Programming"
14      expertiseAreaList=[<ExpertiseArea id=98 name="Java Development" description="Java Development"/> ]/>
15   </Functions>
16   <Processes>
17     <Process id=110 name="Requirement Analysis" description="Requirement Analysis"
18     teamList=[ <Team id=101 name="Requirement Management Team" description="Requirement Management Team" />]
19     inputEntityList=[]
20     outputEntityList=["Requirement Description Document" ]
21     functionList=["Requirement Derivation" ]
22   />
23     <Process id=111 name="Architecture Design" description="Architecture Design"
24     teamList=[ <Team id=102 name="Architecture Team" description="Architecture Team" />]
25     inputEntityList=["Requirement Description Document" ]
26     outputEntityList=["Architecture Description Document" ]
27     functionList=["Architecture Design" ]
28   />
29     <Process id=112 name="Implementation 1" description="Implementation 1"
30     teamList=[ <Team id=103 name="Development Team A" description="Development Team A" />]
31     inputEntityList=["Architecture Description Document" ]
32     outputEntityList=["Software Unit 1" ]
33     functionList=["Java Programming" ]
34   />
35     <Process id=113 name="Implementation 2" description="Implementation 2"
36     teamList=[ <Team id=105 name="Development Team B" description="Development Team B" />]
37     inputEntityList=["Architecture Description Document" ]
38     outputEntityList=["Software Unit 2" ]
39     functionList=["Java Programming" ]
40   />
41 </Processes>
42 </Model>

```

Figure 7.6 GSPML: Example Case

7.3.3 Global Software Coordination Modeling Language (GSCOML)

Textual concrete syntax of Coordination viewpoint is defined as Global Software Coordination Modeling Language. This language is given in Figure 7.7.

```

1 grammar org.xtext.example.mydsl.GSCOML with org.eclipse.xtext.common.Terminals
2
3 generate gSCOML "http://www.xtext.org/example/mydsl/GSCOML"
4
5 CSCOModel:
6     '<Model' 'id=' id=INT 'name=' name=STRING 'creationDate=' creationDate=STRING
7         'lastUpdateDate=' lastUpdateDate=STRING '>'
8         '<Coordinations>'
9         (coordinations+=Coordination) *
10        '</Coordinations>'
11        '<Teams>'
12        (teams+=Team) *
13        '</Teams>'
14    '</Model>';
15
16 Coordination:
17     '<Coordination' 'id=' id=INT 'name=' name=STRING 'description=' description=STRING
18     'coordinators=[' (coordinators+=[ Team|STRING] ) * ']'
19     'coordinatees=[' (coordinatees+=[ Team|STRING] ) * ']'
20     '/>';
21
22 Team:
23     '<Team' 'id=' id=INT 'name=' name=STRING 'description=' description=STRING '/>';

```

Figure 7.7 Global Software Coordination Modeling Language

This language is relatively simpler compared to the other language definitions since only Team and Coordination entities are included. As you can realize, Teams are separated into two as coordinators and coordinatees in the Coordination rule.

The coordination activities given as example in Coordination viewpoint section is expressed in this language in Figure 7.8.

```

1 <Model id=68 name="deneme" creationDate="23.51.2011 17:51" lastUpdateDate="24.51.2011 13:43">
2     <Coordinations>
3         <Coordination id=119 name="Requirements Coordination" description="Requirements Coordination"
4             coordinators=["Requirement Management Team" ]
5             coordinatees=["Architecture Team" ]
6         />
7         <Coordination id=120 name="Architecture Coordination" description="Architecture Coordination"
8             coordinators=["Architecture Team" ]
9             coordinatees=["Development Team A" "Development Team B" ]
10        />
11    </Coordinations>
12    <Teams>
13        <Team id=101 name="Requirement Management Team" description="Requirement Management Team" />
14        <Team id=102 name="Architecture Team" description="Architecture Team" />
15        <Team id=103 name="Development Team A" description="Development Team A" />
16        <Team id=105 name="Development Team B" description="Development Team B" />
17    </Teams>
18 </Model>

```

Figure 7.8 GSCOML: Example Case

7.3.4 Global Software Data Modeling Language (GSDAML)

Textual expression of Data viewpoint is achieved through Global Software Data Modeling Language. You can find the language in Figure 7.9.

```
1 grammar org.xtext.example.mydsl.GSDAML with org.eclipse.xtext.common.Terminals
2
3 generate gSDAML "http://www.xtext.org/example/mydsl/GSDAML"
4
5 GSDAModel:
6     '<Model' 'id=' id=INT 'name=' name=STRING 'creationDate=' creationDate=STRING
7         'lastUpdateDate=' lastUpdateDate=STRING '>'
8         '<Sites>'
9             (sites+=Site) *
10        '</Sites>'
11        '<Teams>'
12            (teams+=Team) *
13        '</Teams>'
14    '</Model>';
15
16 Site:
17     '<Site' 'id=' id=INT 'name=' name=STRING 'description=' description=STRING
18         'location=' location=STRING
19         'dataStorageList=[' (dataStorages+=DataStorage) * ']'
20     '>';
21
22 DataStorage:
23     '<DataStorage' 'id=' id=INT 'name=' name=STRING 'description=' description=STRING
24         'owner=' owner=[Team|STRING] ('parent=' parent=[DataStorage|STRING])?
25         'canStoreDigital=' canStoreDigital?='digital'?
26         'actualFormatList=[' (actualFList+=ActualFormat) * ']'
27         ('digitalFormatList=[' (digitalFList+=DigitalFormat) * '])?
28         'dataEntityList=[' (dataEntities+=DataEntity) * ']'
29     '>';
30
31 Team:
32     '<Team' 'id=' id=INT 'name=' name=STRING 'description=' description=STRING '>';
33
34 DataEntity:
35     '<DataEntity' 'id=' id=INT 'name=' name=STRING 'description=' description=STRING
36         ('parent=' parent=[DataEntity|STRING])?
37         'isDigital=' isDigital?='digital'? 'size=' size=STRING 'sizeType=' sizeType=STRING
38         'creationDate=' creationDate=STRING 'lastUpdateDate=' lastUpdateDate=STRING
39         'actualFormat=' actualFormat=ActualFormat
40         ('digitalFormat=' digitalFormat=DigitalFormat)?
41         'languageList=[' (languageList+=Language) * ']'
42     '>';
43
44 ActualFormat:
45     '<ActualFormat' 'id=' id=INT 'name=' name=STRING '>';
46
47 DigitalFormat:
48     '<DigitalFormat' 'id=' id=INT 'name=' name=STRING 'description=' description=STRING '>';
49
50 Language:
51     '<Language' 'id=' id=INT 'name=' name=ID '>';
```

Figure 7.9 Global Software Data Modeling Language

The start rule is GSDAModel. This rule consists of Site and Team lists. Each Site has Data Storages associated with it and Data Entities are located in these Data Storages.

Figure 7.10 shows New York site of the example given in Data viewpoint section. We didn't include the entire example in textual form here since it is very long.

```

1 <Model id=68 name="deneme" creationDate="23.51.2011 17:51" lastUpdateDate="24.51.2011 14:15">
2   <Sites>
3     <Site id=75 name="New York" description="New York" location="United States"
4       dataStorageList=[
5         <DataStorage id=135 name="New York Disk" description="New York Disk"
6           owner="Requirement Management Team"
7           canStoreDigital=digital
8           actualFormatList=[
9             <ActualFormat id=69 name="Video" />
10            <ActualFormat id=70 name="Sound" />
11            <ActualFormat id=71 name="Text" />
12            <ActualFormat id=72 name="Picture" />
13            <ActualFormat id=73 name="Complex" />
14          ]
15          digitalFormatList=[
16            <DigitalFormat id=123 name=".doc" description=".doc"/>
17            <DigitalFormat id=124 name=".java" description=".java"/>
18          ]
19          dataEntityList=[
20            <DataEntity id=114 name="Requirement Description Document"
21              description="Requirement Description Document" isDigital=digital size="120.0"
22              sizeType="pages" creationDate="10.11.2010" lastUpdateDate="10.11.2010"
23              actualFormat=<ActualFormat id=73 name="Complex" />
24              digitalFormat=<DigitalFormat id=123 name=".doc" description=".doc"/>
25              languageList=[
26                <Language id=81 name=English/>
27              ]
28            />
29          ]
30        />
31      ]
32    />
33  </Sites>
34  <Teams>
35    <Team id=101 name="Requirement Management Team" description="Requirement Management Team" />
36  </Teams>
37 </Model>

```

Figure 7.10 GSDAML: Example Case

7.3.5 Global Software Communication Modeling Language (GSCML)

Data viewpoint's textual concrete syntax is defined as Global Software Communication Language. The language is shown in Figure 7.11.

```

1 grammar org.xtext.example.mydsl.GSCML with org.eclipse.xtext.common.Terminals
2
3 generate gSCML "http://www.xtext.org/example/mydsl/GSCML"
4
5 GSCModel:
6     '<Model' 'id=' id=INT 'name=' name=STRING 'creationDate=' creationDate=STRING
7         'lastUpdateDate=' lastUpdateDate=STRING '>'
8         '<Communications>'
9             (communications+=Communication) *
10        '</Communications>'
11        '<CommunicationPlatforms>'
12            (cPlatforms+=CPlatform) *
13        '</CommunicationPlatforms>'
14        '<Teams>'
15            (teams+=Team) *
16        '</Teams>'
17        '<DataEntities>'
18            (dataEntities+=DataEntity) *
19        '</DataEntities>'
20        '<Processes>'
21            (processes+=Process) *
22        '</Processes>'
23    '</Model>';
24
25 Team:
26     '<Team' 'id=' id=INT 'name=' name=STRING 'description=' description=STRING
27         'isVirtual=' isVirtual?='virtual'? 'isTemporary=' isTemporary?='temporary'? '>';
28
29 Process:
30     '<Process' 'id=' id=INT 'name=' name=STRING 'description=' description=STRING '>';
31
32 CPlatform:
33     '<CommunicationPlatform' 'id=' id=INT 'name=' name=STRING 'description=' description=STRING '>';
34
35 DataEntity:
36     '<DataEntity' 'id=' id=INT 'name=' name=STRING 'description=' description=STRING
37         'isDigital=' isDigital?='digital'? 'size=' size=STRING 'sizeType=' sizeType=STRING
38         'creationDate=' creationDate=STRING 'lastUpdateDate=' lastUpdateDate=STRING
39     '>';
40
41 Communication:
42     '<Communication' 'id=' id=INT 'name=' name=STRING 'description=' description=STRING
43         'isFormal=' isFormal?='formal'? 'isSynchronous=' isSynchronous?='synchronous'?
44         'type=' type=STRING 'isChannel=' isChannel?='channel'? 'timezone=' timezone=STRING
45         'startTime=' startTime=STRING 'finishTime=' finishTime=STRING
46         'communicationPlatform=' platform=[CPlatform|STRING]
47         'callerTeamList=[' (callerTeamList+=[Team|STRING]) * ']'
48         'receiverTeamList=[' (receiverTeamList+=[Team|STRING]) * ']'
49         'dataEntityList=[' (dataEntityList+=[DataEntity|STRING]) * ']'
50         'processList=[' (processList+=[Process|STRING]) * ']'
51     '>';

```

Figure 7.11 Global Software Communication Modeling Language

Communication Platform entity is mapped to the rule CPlatform. This abbreviation is because of the problems of integration of XText with Graphical Modeling Framework.

The example given in Communication viewpoint section is expressed in textual form in Figure 7.12.

```

1 <Model id=68 name="deneme" creationDate="23.51.2011 17:51" lastUpdateDate="24.51.2011 16:01">
2   <Communications>
3     <Communication id=143 name="Email Communication 1" description="Email Communication 1"
4       isFormal= isSynchronous= type="Email" isChannel=
5       timezone="America/Los_Angeles" startTime="30.12.2010" finishTime="30.12.2010"
6       communicationPlatform="Communication Platform 1"
7       callerTeamList=["Architecture Team" ]
8       receiverTeamList=["Requirement Management Team" ]
9       dataEntityList=["Mail Body 1" ]
10      processList=["Architecture Design" ]
11    />
12    <Communication id=144 name="Email Communication 2" description="Email Communication 2"
13      isFormal= isSynchronous= type="Email" isChannel=
14      timezone="America/Los_Angeles" startTime="31.12.2010" finishTime="31.12.2010"
15      communicationPlatform="Communication Platform 2"
16      callerTeamList=["Requirement Management Team" ]
17      receiverTeamList=["Architecture Team" ]
18      dataEntityList=["Mail Body 2" ]
19      processList=["Architecture Design" ]
20    />
21    <Communication id=145 name="Email Communication 3" description="Email Communication 3"
22      isFormal= isSynchronous= type="Email" isChannel=
23      timezone="America/Los_Angeles" startTime="01.01.2011" finishTime="01.01.2011"
24      communicationPlatform="Communication Platform 1"
25      callerTeamList=["Architecture Team" ]
26      receiverTeamList=["Requirement Management Team" ]
27      dataEntityList=["Mail Body 3" ]
28      processList=["Architecture Design" ]
29    />
30    <Communication id=146 name="Email Communication 4" description="Email Communication 4"
31      isFormal= isSynchronous= type="Email" isChannel=
32      timezone="America/Los_Angeles" startTime="02.01.2011" finishTime="02.01.2011"
33      communicationPlatform="Communication Platform 2"
34      callerTeamList=["Requirement Management Team" ]
35      receiverTeamList=["Architecture Team" ]
36      dataEntityList=["Mail Body 4" ]
37      processList=["Architecture Design" ]
38    />
39    <Communication id=147 name="Phone Call" description="Phone Call"
40      isFormal= isSynchronous= type="Phone" isChannel=
41      timezone="America/Los_Angeles" startTime="09.01.2011" finishTime="09.01.2011"
42      communicationPlatform="Communication Platform 1"
43      callerTeamList=["Architecture Team" ]
44      receiverTeamList=["Requirement Management Team" ]
45      dataEntityList=["Speech" ]
46      processList=["Architecture Design" ]
47    />
48  </Communications>
49  <CommunicationPlatforms>
50    <CommunicationPlatform id=141 name="Communication Platform 1" description="Communication Platform 1"/>
51    <CommunicationPlatform id=142 name="Communication Platform 2" description="Communication Platform 2"/>
52  </CommunicationPlatforms>
53  <Teams>
54    <Team id=102 name="Architecture Team" description="Architecture Team" isVirtual= isTemporary= />
55    <Team id=101 name="Requirement Management Team" description="Requirement Management Team" isVirtual= isTemporary= />
56  </Teams>
57  <DataEntities>
58    <DataEntity id=136 name="Mail Body 1" description="Mail Body 1" isDigital=digital size="1.0" sizeType="pages"
59      creationDate="30.12.2010" lastUpdateDate="30.12.2010"/>
60    <DataEntity id=137 name="Mail Body 2" description="Mail Body 2" isDigital=digital size="1.0" sizeType="pages"
61      creationDate="31.12.2010" lastUpdateDate="31.12.2010"/>
62    <DataEntity id=138 name="Mail Body 3" description="Mail Body 3" isDigital=digital size="1.0" sizeType="pages"
63      creationDate="01.01.2011" lastUpdateDate="01.01.2011"/>
64    <DataEntity id=139 name="Mail Body 4" description="Mail Body 4" isDigital=digital size="1.0" sizeType="pages"
65      creationDate="02.01.2011" lastUpdateDate="02.01.2011"/>
66    <DataEntity id=140 name="Speech" description="Speech" isDigital=digital size="3.0" sizeType="megabytes"
67      creationDate="09.01.2011" lastUpdateDate="09.01.2011"/>
68  </DataEntities>
69  <Processes>
70    <Process id=111 name="Architecture Design" description="Architecture Design"/>
71  </Processes>
72 </Model>

```

Figure 7.12 GSCML: Example Case

7.3.6 Global Software Tool Modeling Language (GSTML)

Global Software Tool Modeling Language corresponds to the textual concrete syntax definition of Tool viewpoint. Figure 7.13 includes this language's definition.

```

1 grammar org.xtext.example.mydsl.GSTML with org.eclipse.xtext.common.Terminals
2
3 generate gSTML "http://www.xtext.org/example/mydsl/GSTML"
4
5 GSTModel:
6     '<Model' 'id=' id=INT 'name=' name=STRING 'creationDate=' creationDate=STRING
7         'lastUpdateDate=' lastUpdateDate=STRING '>'
8         '<CommunicationPlatforms>'
9             (cPlatforms+=CPlatform) *
10        '</CommunicationPlatforms>'
11        '<FunctionPlatforms>'
12            (fPlatforms+=FPlatform) *
13        '</FunctionPlatforms>'
14        '<Functions>'
15            (functions+=Function) *
16        '</Functions>'
17        '<Teams>'
18            (teams+=Team) *
19        '</Teams>'
20    '</Model>';
21
22 Team:
23     '<Team' 'id=' id=INT 'name=' name=STRING 'description=' description=STRING '>';
24
25 CPlatform:
26     '<CommunicationPlatform' 'id=' id=INT 'name=' name=STRING 'description=' description=STRING
27         'owner=' ownerTeam=[Team] STRING
28         'toolList=[' (toolList+=Tool) * ']'
29     '>';
30
31 FPlatform:
32     '<FunctionPlatform' 'id=' id=INT 'name=' name=STRING 'description=' description=STRING
33         'owner=' ownerTeam=[Team] STRING
34         'toolList=[' (toolList+=Tool) * ']'
35     '>';
36
37 Tool:
38     '<Tool' 'id=' id=INT 'name=' name=STRING 'description=' description=STRING
39         'supportCollaboration=' sCollaboration?='collaboration'?
40         'actualFormatList=[' (actualFList+=ActualFormat) * ']'
41         'digitalFormatList=[' (digitalFList+=DigitalFormat) * ']'
42     '>';
43
44 ActualFormat:
45     '<ActualFormat' 'id=' id=INT 'name=' name=STRING '>';
46
47 DigitalFormat:
48     '<DigitalFormat' 'id=' id=INT 'name=' name=STRING 'description=' description=STRING '>';
49
50 Function:
51     '<Function' 'id=' id=INT 'name=' name=STRING 'description=' description=STRING
52         'platform=' platform=[FPlatform] STRING '>';

```

Figure 7.13 Global Software Tool Modeling Language

Like Communication Platform entity, Function Platform entity is also mapped to the rule FPlatform to solve the problems of integration of XText with Graphical Modeling Framework.

The language instance given in Figure 7.14 represents the example presented in Tool viewpoint section.

```

1 <Model id=68 name="deneme" creationDate="23.51.2011 17:51" lastUpdateDate="24.51.2011 18:22">
2   <CommunicationPlatforms>
3 </CommunicationPlatforms>
4   <FunctionPlatforms>
5     <FunctionPlatform id=148 name="Function Platform 1" description="Function Platform 1" owner="Requirement Management Team"
6       toolList=[
7         <Tool id=152 name="Doors" description="Requirement Management Tool" supportCollaboration=collaboration
8           actualFormatList=[
9             <ActualFormat id=69 name="Video" /> <ActualFormat id=70 name="Sound" />
10            <ActualFormat id=71 name="Text" /> <ActualFormat id=72 name="Picture" />
11            <ActualFormat id=73 name="Complex" />
12          ]
13          digitalFormatList=[
14            <DigitalFormat id=123 name=".doc" description=".doc"/>
15          ]
16        ]
17      ]
18    ]
19    <FunctionPlatform id=149 name="Function Platform 2" description="Function Platform 2" owner="Architecture Team"
20      toolList=[
21        <Tool id=153 name="Enterprise Architect" description="Enterprise Architect" supportCollaboration=
22          actualFormatList=[
23            <ActualFormat id=69 name="Video" /> <ActualFormat id=70 name="Sound" />
24            <ActualFormat id=71 name="Text" /> <ActualFormat id=72 name="Picture" />
25            <ActualFormat id=73 name="Complex" />
26          ]
27          digitalFormatList=[
28            <DigitalFormat id=124 name=".java" description=".java"/>
29          ]
30        ]
31      ]
32    ]
33    <FunctionPlatform id=150 name="Function Platform 3" description="Function Platform 3" owner="Development Team A"
34      toolList=[
35        <Tool id=154 name="Eclipse IDE" description="Eclipse IDE" supportCollaboration=collaboration
36          actualFormatList=[
37            <ActualFormat id=69 name="Video" /> <ActualFormat id=70 name="Sound" />
38            <ActualFormat id=71 name="Text" /> <ActualFormat id=72 name="Picture" />
39            <ActualFormat id=73 name="Complex" />
40          ]
41          digitalFormatList=[
42            <DigitalFormat id=124 name=".java" description=".java"/>
43          ]
44        ]
45      ]
46    ]
47  </FunctionPlatforms>
48  <Functions>
49    <Function id=107 name="Requirement Derivation" description="Requirement Derivation" platform="Function Platform 1"/>
50    <Function id=108 name="Architecture Design" description="Architecture Design" platform="Function Platform 2"/>
51    <Function id=109 name="Java Programming" description="Java Programming" platform="Function Platform 3"/>
52  </Functions>
53  <Teams>
54    <Team id=101 name="Requirement Management Team" description="Requirement Management Team" />
55    <Team id=102 name="Architecture Team" description="Architecture Team" />
56    <Team id=103 name="Development Team A" description="Development Team A" />
57  </Teams>
58 </Model>

```

Figure 7.14 GSTML: Example Case

Because of the space limitation, we omitted Function Platform 4 which is same with Function Platform 4.

7.3.7 Global Software Migration Modeling Language (GSMML)

Global Software Migration Modeling Language is given in Figure 7.15. This language is the textual representation of Migration viewpoint.

```

1 grammar org.xtext.example.mydsl.GSMML with org.eclipse.xtext.common.Terminals
2
3 generate gSMML "http://www.xtext.org/example/mydsl/GSMML"
4
5 GSMModel:
6     '<Model' 'id=' id=INT 'name=' name=STRING 'creationDate=' creationDate=STRING
7         'lastUpdateDate=' lastUpdateDate=STRING '>'
8         '<Migrations>'
9         (migrations+=Migration) *
10        '</Migrations>'
11        '<Sites>'
12        (sites+=Site) *
13        '</Sites>'
14    '</Model>';
15
16 Team:
17     '<Team' 'id=' id=INT 'name=' name=STRING 'description=' description=STRING '>';
18
19 Process:
20     '<Process' 'id=' id=INT 'name=' name=STRING 'description=' description=STRING '>';
21
22 DataStorage:
23     '<DataStorage' 'id=' id=INT 'name=' name=STRING 'description=' description=STRING
24         'canStoreDigital=' canStoreDigital?='digital'? '>';
25
26 Site:
27     '<Site' 'id=' id=INT 'name=' name=STRING 'description=' description=STRING 'location=' location=STRING '>';
28
29 Migration:
30     '<Migration' 'id=' id=INT 'name=' name=STRING 'description=' description=STRING 'date=' date=STRING
31         'fromSite=' fromSite=[Site|STRING] 'toSite=' toSite=[Site|STRING]
32         'teamList=[' (teamList+=Team) * ']'
33         'processList=[' (processList+=Process) * ']'
34         'dataStorageList=[' (dataStorageList+=DataStorage) * ']'
35     '>';

```

Figure 7.15 Global Software Migration Modeling Language

Like the visual presentation, the textual model has also Migration and Site lists in starting rule. Process, Data Storage and Team entities are included in Migration entity.

Migration activity given as example in Migration viewpoint section is expressed in textual form in Figure 7.16.

```

1 <Model id=68 name="deneme" creationDate="23.51.2011 17:51" lastUpdateDate="24.51.2011 19:14">
2   <Migrations>
3     <Migration id=156 name="Migration" description="Migration" date="10.12.2010"
4       fromSite="New York" toSite="California"
5       teamList=[
6         <Team id=101 name="Requirement Management Team"
7           description="Requirement Management Team" />
8       ]
9       processList=[
10        <Process id=110 name="Requirement Analysis" description="Requirement Analysis"/>
11        <Process id=111 name="Architecture Design" description="Architecture Design"/>
12      ]
13      dataStorageList=[
14        <DataStorage id=155 name="External Disk" description="External Disk"
15          canStoreDigital=digital
16        />
17      ]
18    />
19  </Migrations>
20  <Sites>
21    <Site id=76 name="California" description="California" location="United States"/>
22    <Site id=75 name="New York" description="New York" location="United States"/>
23  </Sites>
24 </Model>

```

Figure 7.16 GSMML: Example Case

7.4 Using Graphical Modeling Framework to Define Visual Representations

After the definition of the textual concrete syntaxes of the viewpoints of GSD in XText, we have textual presentation's tool support for the viewpoints. We can model our application architecture from different viewpoints in textual form using domain-specific editor and also the same editor validates our textual model depending on the syntactic definition of viewpoint's language. Now, we want the same support for the visual representation of viewpoints, too. For this, we have used Eclipse Graphical Modeling Framework.

Eclipse Graphical Modeling Framework [7] provides a model-driven approach with the aim of generating graphical editors for domain-specific languages. The first step in generating graphical editors is the definition of domain-specific meta-model in ECore form. ECore is a meta meta-model definition in the content of Eclipse Modeling Framework [8] to allow designers to design their own meta-models. XText tool automatically derives the ECore model from the syntactic language definition.

The second step is the definition of the intermediate models. The intermediate models consist of Graphical Definition (*.gmfgraph), Tooling Definition (*.gmftool), Mapping Model (*.gmfmap) and Generator models (*.gmfgen). Graphical Definition model is used for defining the figures, nodes, links between these nodes, etc. that are used in visualization in model diagram. Tooling model is for specifying the palette, creation tools, actions, etc. for the graphical elements. Mapping model functions as a linker between the ECore model, Graphical Definition model and Tooling model. After this linking phase, GMF generates Generator model which is used for creating the graphical editor itself.

The intermediate model generation becomes a hard work if it is wished to do manually since it each model has its own meta-model and to generate good intermediate models, one should have a deep understanding of these complex meta-models. To cope with these difficulties, we used Eugenia [9] which is a tool designed for raise the abstraction level of GMF. Eugenia creates the intermediate models by itself by using the textual version of the ECore model. The Ecore model is annotated using suitable annotations with the desired visual notations and by interpreting these annotations, Eugenia generates the intermediate models.

As an example, we present the annotated version of the ECore model of Migration viewpoint in Figure 7.17.

```

1 @namespace(uri="http://www.xtext.org/example/mydsl/GSMM", prefix="GSMM")
2 @gmf(foo="bar")
3 package GSMM;
4
5 @gmf.diagram(foo="bar")
6 class GSMMModel {
7   attr int ~id;
8   attr String name;
9   attr String creationDate;
10  attr String lastUpdateDate;
11  val Migration[*] migrations;
12  val Site[*] sites;
13 }
14
15 @gmf.node(figure="figures.Team", label="name", label.icon="false", label.placement="external", tool.name="Team")
16 class Team {
17   attr int ~id;
18   attr String name;
19   attr String description;
20 }
21
22 @gmf.node(label="name", label.icon="true", label.placement="internal", figure="rounded", tool.name="Process")
23 class Process {
24   attr int ~id;
25   attr String name;
26   attr String description;
27 }
28
29 @gmf.node(figure="figures.DataStorage", label="name", label.icon="false", label.placement="external", tool.name="Data Storage")
30 class DataStorage {
31   attr int ~id;
32   attr String name;
33   attr String description;
34   attr boolean canStoreDigital;
35 }
36
37 @gmf.node(label="name", label.icon="true", label.placement="internal", figure="rounded", tool.name="Site")
38 class Site {
39   attr int ~id;
40   attr String name;
41   attr String description;
42   attr String location;
43 }
44
45 @gmf.node(label="name", label.icon="true", label.placement="internal", figure="rounded", tool.name="Migration")
46 class Migration {
47   attr int ~id;
48   attr String name;
49   attr String description;
50   attr String date;
51   @gmf.link(source.decoration="filledclosedarrow", width="2")
52   ref Site fromSite;
53   @gmf.link(target.decoration="filledclosedarrow", width="2")
54   ref Site toSite;
55   @gmf.compartment(layout="free", collapsible="false")
56   val Team[*] teamList;
57   @gmf.compartment(layout="free", collapsible="false")
58   val Process[*] processList;
59   @gmf.compartment(layout="free", collapsible="false")
60   val DataStorage[*] dataStorageList;
61 }

```

Figure 7.17 Annotated Textual ECore Model of Migration Viewpoint

On the second line, the first annotation appears. This annotation declares that this model file includes annotations. On the fifth line, the diagram is declared. It means that GSMMModel class will be modeled as diagram. On the lines before the class declarations, the annotations show that how the classes will be visualized in the diagrams. Team and Data Storage classes will be represented by pictures (the pictures given in the viewpoint definition. Process, Site and Migration classes will be represented in box forms as in UML class diagrams. In addition to these, Migration class's attributes are represented in the figure. The source Site from which Migration occurs (fromSite attribute) is represented with a link incoming from source Site and the target Site is represented with an

outgoing link from Migration class. Team, Process and Data Storage lists are represented in containment form inside Migration class.

For each of the viewpoints, we have defined the visual concrete syntaxes. These definitions are then used in the generation of graphical editors using Eugenia and GMF. The annotated Ecore file given above are given as input to Eugenia and Eugenia creates the intermediate models. These intermediate models are used in generation of Migration Viewpoint Graphical Editor by GMF. Then this new editor is ready for both visual modeling from Migration viewpoint. The textual example we have given in Global Software Migration Modeling Language section is modeled as in Figure 7.18 by the graphical editor.

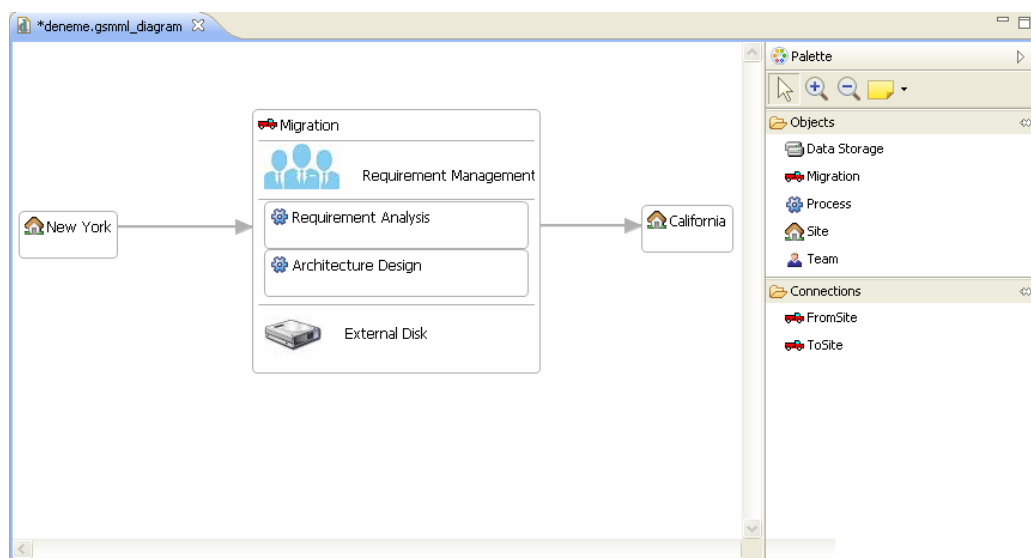


Figure 7.18 Migration Viewpoint Graphical Editor: Example Case

One important point that must be noted is that the process in which the graphical editor is generated, a link between visual editor and textual editor is created. In other words, one can derive the visual diagram from the textual definition automatically by using GMF and at the same time, any change done in the visual diagram is directly reflected to the textual definition.

Here, we have provided Migration Viewpoint Graphical Editor as an example for explaining graphical editor generation process. Since the visual presentation examples given in viewpoint sections look very similar to the graphical editor outputs, we do not further elaborate on the remaining viewpoints.

Chapter 8

Related Work

Architecting GSD is not widely addressed in the literature. The key research focus in the GSE community seems to have focused on the problems related to communication, coordination and control concerns and more social issues. Clerk et al. [38] report on the use of *architectural rules* to cope with challenges in GSD environments. In Clerk's study, *architectural rules* are described as "principles or statements about the software architecture that must be complied with through organization". They list four main challenges according to GSD: time difference and geographical distance, culture, team communication and collaboration, work distribution. For each of these challenges, they propose a collection of solutions and explain to what extent these solutions can be expressed as architectural rules. Our work can be considered as a complementary study to Clerk's work. In our study, we defined design actions and mapped them to expected answers.

Avritzer et al. report their experience in [39] in assessing the relationship between the coordination needs of distributed development teams and the dependency structure of a software architecture. They represent the architectural dependencies and the coordination structure using matrix models in the Global

Studio Project Version 3.0. The analysis of gathered data showed that design structure matrix (DSM) models representing the software architecture of GSD can guide the task assignments in Global Software Development projects.

In our study, the question framework has an important function of derivation the GSD application architecture. The notion of question framework has been addressed in many studies in the literature. Generally, question frameworks have been used for evaluating architectural descriptions, not for supporting the development of the architecture as we proposed. As an example, in [40], Hämäläinen and Markkula offered a question framework for evaluating the quality of architectural descriptions. They organized the quality under four categories: *stakeholder and purpose orientation*, *content quality*, *presentation and visualization quality*, and *management of documents*. The first one, *stakeholder and purpose orientation*, aims to measure how well documents serve their purpose and aim stakeholders using them. The quality of the information included in the documents is evaluated by *content quality* criteria. The *presentation and visualization quality* is defined for evaluating how well information is presented in the documentation. The last criterion, management of documents, focuses on evaluating the quality of architectural description from the viewpoint of processes and practices.

Similarly, Nord et al. [41] presents a structured approach which includes question frameworks for reviewing the architectural documentation. The approach aims to help the documentation creators and organizers to ensure that the architecture documentation meets quality concerns of stakeholders of the target artifact.

Tool support for especially for allowing collaborative development is defined as one of the important challenges in GSD since it requires the development of development tools and supporting collaboration between developers [42]. Booch and Brown have introduced the concept of *Collaborative Development*

Environment (CDE) [43]. In their work, CDE is defined as “a virtual space wherein all the stakeholders of the project - even if distributed by time and distance – may carry out some task, most often to create an executable deliverable and its supporting artifacts”. Many works are proposed to support the idea of CDEs. *Collab.net* [44] is a commercial tool supporting CDE, and offers services of configuration management, bug tracking, task management and discussion. *SoftFab* proposed by Spanjer et al. in [45] automates the build and test processes in multi-site projects. Carroll et al. [46] presents *Jazz*, a tool for supporting rich synchronous communication and raises the mutual awareness of coding activities within a development team.

Except these known examples, many tools are proposed in the literature to support collaborative requirements management, communication and development activities. Whitehead [47] has examined more than fifty existing collaboration support tools and collected them in his survey. He categorizes these tools in four broad categories: *Model-based*, *Process support*, *Awareness tools* and *Collaboration infrastructure*. Whitehead states that collaborative work of creating artifacts is the collaborative work of creating models of the system. So, tools supporting collaborative artifact creation are classified as *Model-based* tools. *Process support* tools aims to manage task assignments of engineers, monitoring activities and invoking other necessary tools based on the processes expressed by process models followed by the stakeholders. *Awareness tools* are used to increase awareness among developers, in other words, making realize what is going on on the other side. This aid engineers to perform coordination activities sooner and so to avoid conflicts and unnecessary rework. *Collaboration infrastructure* solves interoperability problems by supporting data and control integration between different tools and interfaces.

Maciel et al. [48] present a domain-specific architecture (DSA) serving as middleware which provides interoperability in collaborative environments. The reference architecture (Platform Independent Model) is based on Object

Management Group's UML Profile for Enterprise Distributed Object Computing (EDOC) [49] and the viewpoints defined in RM-ODP (Open Distributed Processing-Reference Model) [50] are adopted.

Although existing CDE in the literature tools offer clear benefits, it appears that the literature on CDE focuses on the collaborative concerns and gives less emphasize on development part. Further, the tools addressing development pays little attention to architecture design compared to collaborative coding. Our proposed approach and tool can be considered as a work for enhancing CDE for design of GSD environments.

Chapter 9

Conclusion

Different challenges have been identified to set up a GSD environment. Our literature study showed that GSD community has focused on communication, coordination and control challenges in particular and has paid less attention to modeling, documentation and analysis of architecture for GSD environment. In this thesis, we have focused on the architecture design of GSD environment.

Designing architecture for a single system is hard. Designing architecture for GSD environment is even more difficult because of the additional challenges of communication, coordination and control of distributed GSD teams. To support the GSD architect in designing a proper GSD architecture, we have provided both the approach and the tool support.

We have proposed seven architectural viewpoints for modeling GSD application architecture. In this thesis we consider a viewpoints as a domain specific language or meta-model. Consequently to derive the viewpoints we have first defined a GSD meta-model based on a thorough literature. The meta-model is divided into six units which are also called as language units. These units form the abstract syntaxes of viewpoints. We have completed the

architectural viewpoint definition by defining textual and visual concrete syntaxes.

The question framework that we have presented showed to be useful in deriving architecture application for GSD. To support this process further, we have developed the tool *Global Architect* with which we can create textual and visual models of the architectural application.

In this thesis, we have focused on the modeling of application architecture of GSD from different viewpoints. As a future work, we see different perspectives. First of all, it would be valuable to apply the approach in a real industrial setting in which GSD systems are developed. Further, if needed, the meta-model can be extended to support the definition of other viewpoints. Finally, we think that the approach and the tool can also be used to support the evaluation of the GSD architecture. The evaluation of the architecture could be, for example, based on cost of development which includes task allocations, communication and migration costs and employment of man-force from different countries and so on.

BIBLIOGRAPHY

- 1 P. Clements, F. Bachmann, L. Bass, D. Garlan, J. Ivers, R. Little, P. Merson, R. Nord, J. Stafford. Documenting Software Architectures: Views and Beyond. Second Edition. Addison-Wesley, 2010.
- 2 Software & Systems Process Engineering Meta-Model Specification, Object Management Group (OMG), <http://www.omg.org/spec/SPEM/2.0/PDF>, accessed in June 2011.
- 3 Rational Doors, <http://www-01.ibm.com/software/awdtools/doors/>, accessed in June 2011.
- 4 Enterprise Architect, <http://www.sparxsystems.com/products/ea/index.html>, accessed in June 2011.
- 5 Eclipse Integrated Development Environment, <http://www.eclipse.org/>, accessed in June 2011.
- 6 XText, <http://www.eclipse.org/Xtext/>, accessed in June 2011.
- 7 Graphical Modeling Framework, <http://www.eclipse.org/modeling/gmp/>, accessed in June 2011.
- 8 Eclipse Modeling Framework, <http://www.eclipse.org/modeling/emf/>, accessed in June 2011.

- 9 Eugenia, <http://www.eclipse.org/gmt/epsilon/doc/eugenia/>, accessed in June 2011.
- 10 van Deursen, A., Klint, P., Visser, J., Domain-specific languages: an annotated bibliography, *ACM Sigplan Notices*, 35 (6), 26-36, 2000.
- 11 Bikram Sengupta , Satish Chandra , Vibha Sinha, A research agenda for distributed software development, *Proceedings of the 28th international conference on Software engineering*, May 20-28, 2006, Shanghai, China.
- 12 J.D. Herbsleb, *Global Software Engineering: The Future of Socio-technical Coordination*, 2007 *Future of Software Engineering*, p.188-198, May 23-25, 2007.
- 13 Institute of Electrical and Electronics Engineers. *IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries*. New York, NY: 1990.
- 14 Cost of Living Survey, Mercer Press Office. <http://www.mercer.com/costoflivingpr>. Accessed in June 19, 2011.
- 15 Carmel, E., Agarwal, R.: Tactical Approaches for Alleviating Distance in Global Software Development. *IEEE Software* 18(2), 22–29 (2001)
- 16 PayScale. <http://www.payscale.com>. Accessed in June 20, 2011.
- 17 Grinter, R.E., Herbsleb, J.D., Perry, D.E.: The Geography of Coordination: Dealing with Distance in R&D Work. In: *International Conference on Supporting Group Work 1999*, pp. 306–315 (1999).
- 18 Ebert, C., De Neve, P.: Surviving Global Software Development. *IEEE Software* 18(2), 62–69 (2001).
- 19 Carmel, E.: *Global Software Teams: Collaborating Across Borders and Time Zones*, 1st edition. Prentice-Hall, Upper Saddle River (1999).
- 20 Ebert, C., Parro, C.H., Suttels, R., Kolarczyk, H.: Improving validation activities in a global software development. In: *Proceedings of the 23rd International Conference on Software Engineering*, Toronto, Canada (2001).

- 21 Pär J. Ågerfalk , Brian Fitzgerald , Helena Holmström Olsson , Eoin Ó Conchúir, Benefits of Global Software Development: The Known and Unknown, Proceedings of the Software process, 2008 international conference on Making globally distributed software development a success story, May 10-11, 2008, Leipzig, Germany.
- 22 Ó Conchúir, E, Holmström, H., Fitzgerald, B., and Ågerfalk, P.J. (2006). Global software development: Never Mind the Problems – Are There Really Any Benefits? In Proceedings of IRIS (Information Systems Research in Scandinavia), Helsingør, Denmark, 12-15 August, 2006.
- 23 Galbraith, J. (1977). Organizational design. Reading, MA: Addison-Wesley.
- 24 Kraut, R. E. & Streeter., L. A. (1995). Coordination in software development. Communications of the ACM, 38(3), 69-81.
- 25 M. O'Hara-Devereaux and B. Johansen, Global Work: Bridging Distance, Culture, and Time. San Francisco: Jossey-Bass, 1994.
- 26 C. Handy, "Trust and the Virtual Organization", Harvard Business Review, volume 73, pp. 40-50, 1995.
- 27 A. Mockus , J. Herbsleb, Challenges of Global Software Development, Proceedings of the 7th International Symposium on Software Metrics, p.182, April 04-06, 2001.
- 28 C. Hofmeister, R. Nord, and D. Soni. Applied Software Architecture. Addison-Wesley, NJ, USA.
- 29 P. Kruchten. The 4+1 View Model of Architecture. IEEE Software, 12(6):42–50, 1995.
- 30 N. Rozanski and E.Woods. Software Systems Architecture – Working with Stakeholders using Viewpoints and Perspectives. Addison-Wesley, 2005.
- 31 [ISO/IEC 42010:2007] Recommended practice for architectural description of software-intensive systems (ISO/IEC 42010). (identical to ANSI/IEEE Std1471–2000), July 2007.

- 32 E. Demirli & B. Tekinerdogan. Software Language Engineering of Architectural Viewpoints, in Proc. of the 5th European Conference on Software Architecture (ECSA 2011), LNCS 6903, pp. 336–343, 2011.
- 33 J. Bézivin. On the Unification Power of Models. *Software and System Modeling (SoSym)* 4(2):171-188, 2005.
- 34 A. Kleppe. *Software Language Engineering: Creating Domain-Specific Languages Using Metamodels*. Addison-Wesley Longman Publishing Co., Inc., Boston, 2009.
- 35 [ISO/IEC 10746-2:1996] *Information Technology - Open Distributed Processing - Reference Model: Foundations (ISO/IEC 10746-2)*. 1996.
- 36 TOGAF 1995 -The Open Group Architecture Framework, Version 8.1.1., 1995. <http://www.opengroup.org/architecture/togaf8-doc/arch/>
- 37 J.A. Zachman. A Framework for Information Systems Architecture. *IBM Systems Journal*, Vol. 26. No 3, pp. 276-292, 1987.
- 38 V. Clerc, P. Lago and H. van Vliet, “Global Software Development: Are Architectural Rules the Answer?” Proc. of the 2nd International Conference on Global Software Engineering, pp. 225–234. IEEE Computer Society Press, Los Alamitos, 2007.
- 39 A. Avritzer, D. Paulish, Y. Cai. Coordination Implications of Software Architecture in a Global Software Development Project, Seventh Working IEEE/IFIP Conference on Software Architecture, pp. 107-116, 2008.
- 40 Hämäläinen, N. & Markkula, J. —Quality Evaluation Question Framework

for Assessing the Qual-ity of Architecture Documentation.|| Proceedings of

International BCS Conference on Software Quality Management. University of Tampere, SQM, 2007.

- 41 R. Nord, P. Clements, D. Emery, R. Hilliard. A Structured Approach for Reviewing Architecture Documentation, Technical Note, CMU/SEI-2009-TN-0302009, SEI-CMU, 2009.
- 42 B. Sengupta, S. Chandra, V. Sinha. A research agenda for distributed software development, In Proceedings of the 28th international conference on Software engineering, pp. 731-740, 2006.
- 43 G. Booch and A. Brown. *Collaborative Development Environments*. Advances in Computers Vol. 59, Academic Press, August, 2003.
- 44 Collab.net, <http://www.collab.net>, <http://sourceforge.net>, accessed in August, 2011.
- 45 H. Spanjers, Ter, B. Graaf, M. Lormans, D. Bendas, R. V. Solingen. Tool Support for Distributed Software Engineering, in Proc of: International Conference on Global Software Engineering, 2006. ICGSE '06, pp. 187-198, 2006.
- 46 M. Carroll and S. Sprenkle. *Coven: Brewing Better Collaboration through Software Configuration Management*. ACM SIGSOFT Foundations of Software Engineering, pp. 88-97, 2000.
- 47 J. Whitehead, *Collaboration in Software Engineering: A Roadmap*, In FOSE '07: 2007 Future of Software Engineering, pp. 214-225, 2007.
- 48 R. S. P. Maciel, C. G. Ferraz, and N. S. Rosa, “An MDA domain specific architecture to provide interoperability among collaborative environments,” in Proceedings of the 19th Brazilian Symposium on Software Engineering (SBES '05), pp. 1–16, Uberlandia, Brazil, October 2005.
- 49 OMG EDOC. UML Profile for Enterprise Distributed Object Computing Specification. OMG Adopted Specification (ptc/02-02-05), 2002.
- 50 ISO-2004. Use of UML for ODP system specification. Working Draft. ISSO/IEC JTC1/SC7.

- 51 D. M. German, The GNOME Project: A Case Study of Open Source, Global Software Development, Software Process Improvement and Practice. 2003; 8: 201–215.
- 52 S.J. Mellor, K. Scott, A. Uhl, D. Weise. MDA Distilled: Principle of Model Driven Architecture, Addison Wesley, Reading , 2004.

Publications Produced in This Thesis

- 1 B. M. Yildiz & B. Tekinerdogan. Architectural Viewpoints for Global Software Development. in the Proc. of Global Software Engineering, First International Workshop on Architecting in Helsinki, pp. 9-16, August, 2011.
- 2 B. M. Yildiz & B. Tekinerdogan. Meta-Model for Global Software Development to Support Portability and Interoperability, accepted for publication in The Sixth International Conference on Software Engineering Advances (ICSEA 2011), Barcelona, Spain, October, 2011.
- 3 B. M. Yildiz & B. Tekinerdogan. Küresel Yazılım Geliştirme Projeleri İçin Alana Özgü Dil, accepted for publication in the 5. Ulusal Yazılım Mimarisi Sempozyumu (UYMS'11), Ankara, Turkey, September, 2011.