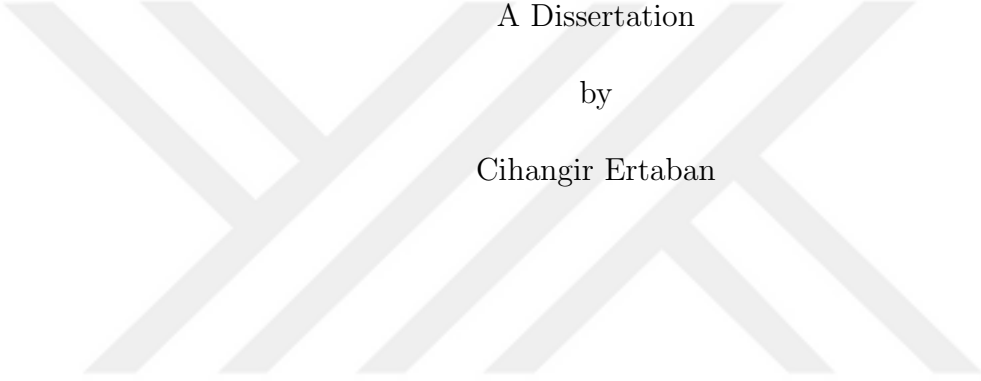


RESOURCE OPTIMIZATION OF SOFTWARE DEVELOPMENT TEAMS WORKING WITH KANBAN



A Dissertation

by

Cihangir Ertaban

Submitted to the
Graduate School of Sciences and Engineering
in Partial Fulfillment of the Requirements for
the Degree of

Doctor of Philosophy

in the
Department of Industrial Engineering

Özyeğin University
June 2024

Copyright © 2024 by Cihangir Ertaban

RESOURCE OPTIMIZATION OF SOFTWARE DEVELOPMENT TEAMS WORKING WITH KANBAN

Approved by:

Assistant Prof. Dr. Erinç ALBEY,
Advisor
Department of Artificial Intelligence
and Data Engineering
Özyeğin University

Assistant Prof. Dr. Mehmet ÖNAL
Department of Industrial Engineering
Özyeğin University

Associate Prof. Dr. İhsan
YANIKOĞLU
Department of Industrial Engineering
Özyeğin University

Assistant Prof. Dr. Görkem YILMAZ
Department of Industrial Engineering
Izmir Economy University

Date Approved: May 17, 2024

Prof. Dr. Mehmet Güray GÜLER
Department of Industrial Engineering
Istanbul Technical University



To my mother..

ABSTRACT

Kanban has been a commonly used agile and lean way of working for software development teams due to its evolutionary approach, effectiveness in project management, and wide range of applicability. This thesis takes a holistic approach to investigate the optimization of software development teams working with Kanban. Due to the complexity of the nature of work, which involves multiple workstations, resources with different skill sets, work types, blocking scenarios, and work-in-process limits, simulations are conducted to gather data. The study proposes a solution framework that approaches the problem at two levels and an additional exploratory multi-objective optimization section using modern techniques to delve deeper into potential directions. The first level of the solution framework starts with a novel, comprehensive simulation algorithm that introduces a blocking scenario applied to the work items, unlike the machine breakdown method used in the manufacturing processes. The simulation model takes input parameters derived from a real-life company, adds resources, and WIP limits as other input parameters and collects data including output, lead time, blocking time, efficiency of resources, and more. This simulation module is then plugged into the solution methods including a greedy heuristic, a two-step model, and a decision tree clearing function to compare the three methods and find the best-performing under different conditions. The second level of the solution framework extends the simulation model by adding resource capabilities and work type priorities ending in over tens of millions of potential combinations. Due to the intractability of generating all possible combinations, three machine learning algorithms -random forest, XGBoost, and neural networks- are used to find the best potential feature set. Finally, a holistic approach is taken to account for lead time, output, and cost

via multi-objective optimization models. Numerical analysis provides insights into the impact of resources, and work-in-progress limits through three strong evolutionary algorithms, which are strength Pareto evolutionary algorithm-II, Pareto archived evolution strategy, and non-dominated sorting genetic algorithm-II. The application of business priorities and the effect of these choices are demonstrated in navigating the trade-offs between cost, lead time, and output. This dissertation contributes significantly to both academic knowledge and practical applications, presenting pioneering insights into optimizing the output and lead time of a Kanban software development team, impacts of resources, work-in-progress limits, and priorities and provides a set of state-of-the-art practical solutions for achieving this goal.

ÖZETÇE

Kanban, evrimsel yaklaşımı, proje yönetimindeki etkinliği ve geniş uygulama alanları nedeniyle yazılım geliştirme takımları tarafından yaygın olarak kullanılan bir çevik ve yalın çalışma yöntemi olmuştur. Bu tez, Kanban kullanan yazılım geliştirme takımlarının optimizasyonunu bütünsel bir yaklaşımla incelemektedir. Çalışma alanı karmaşıklığı, çok sayıda çalışma istasyonunu, farklı beceri setlerine sahip kaynakları, iş türlerini, tıkanma senaryolarını ve işlemdeki sınırlamaları içermesi nedeniyle veri toplamak amacıyla simülasyonlar yapılmaktadır. Çalışma, iki seviyede problemi ele alan bir çözüm çerçevesi ve potansiyel yönleri daha derinlemesine araştırmak için modern teknikler kullanılan ek bir keşifsel çok amaçlı optimizasyon bölümü önermektedir. Çözüm çerçevesinin ilk seviyesi, üretim süreçlerinde kullanılan makine arızası metodundan farklı olarak, iş öğelerine uygulanan bir blok senaryosu tanıtan yenilikçi ve kapsamlı bir simülasyon algoritması ile başlar. Simülasyon modeli, gerçek bir şirketten alınan girdi parametrelerini kullanır, kaynaklar ve WIP sınırlamaları gibi diğer girdi parametrelerini ekler ve çıktı, teslim süresi, blok süresi, kaynakların verimliliği gibi verileri toplar. Bu simülasyon modülü, üç metodun karşılaştırılması ve farklı koşullar altında en iyi performansı göstereni bulmak için bir açgözlü sezgisel, iki aşamalı model ve bir karar ağacı temizleme fonksiyonuna entegre edilir. Çözüm çerçevesinin ikinci seviyesi, simülasyon modelini kaynak yetenekleri ve iş öncelikleri ekleyerek genişletir ve on milyonlarca potansiyel kombinasyonla sonuçlanır. Tüm olası kombinasyonları üretmenin pratik olmaması nedeniyle, en iyi potansiyel özellik setini bulmak için üç makine öğrenimi algoritması -rastgele orman, XGBoost ve yapay sinir ağları- kullanılmaktadır. Son olarak, çok amaçlı optimizasyon modelleri aracılığıyla teslim süresi, çıktı ve maliyet göz önünde bulundurularak bütünsel bir yaklaşım

benimsenir. Sayısal analiz, güçlü üç evrimsel algoritma aracılığıyla kaynakların ve işlemdeki ilerlemenin sınırlarının etkilerini ortaya koyar; bunlar, güç Pareto evrimsel algoritması-II (SPEA-II), Pareto arşivli evrim stratejisi (PAES) ve baskın olmayan sıralama genetik algoritması-II (NSGA-II)'dir. İş önceliklerinin uygulanması ve bu seçimlerin etkileri, maliyet, teslim süresi ve çıktı arasındaki dengelemeleri yönlendirme konusunda gösterilmektedir. Bu tez, hem akademik bilgiye hem de pratik uygulamalara önemli katkılar sağlayarak, bir Kanban yazılım geliştirme takımının çıktısını ve teslim süresini optimize etme, kaynakların, işlemdeki ilerlemenin sınırlamalarının ve önceliklerin etkilerini ortaya koyma ve bu hedefe ulaşmak için son teknoloji pratik çözümler sunma konusunda öncü içgörüler sunmaktadır.

ACKNOWLEDGEMENTS

First of all, I would like to express my sincere gratitude to my advisor Assoc. Prof. Dr. Erinç Albey for his continuous patience, support and guidance at the utmost level. During my study, I faced many personal challenges, and he was always been there to support me. I also would like to thank Assoc. Prof. Dr. İhsan Yanıkoğlu, and Prof. Dr. Mehmet Güray Güler for their feedback, and support as my thesis monitoring committee. In addition, I would like to thank them for participating in my jury, together with Assistant Prof. Dr. Mehmet Önal, and Assistant Prof. Dr. Görkem Yılmaz.

I am deeply thankful to my family, whose constant support and love have been the foundation of my strength. I am especially grateful to my father, from whom I have learned invaluable lessons that have profoundly shaped my character. I thank my sister, Ashhan Demirel, from my heart for her inspiration and guidance, and Burçin Demirel for his endless positivity and selflessness. I also cherish the joy that Ada and Ege, my beloved niece and nephew, bring into my life. They all bring the meaning of family to my life. Lastly, a special thank you to my dear Lydia, who provided endless support during the most challenging times of this thesis.

TABLE OF CONTENTS

DEDICATION	iii
ABSTRACT	iv
ÖZETÇE	vi
ACKNOWLEDGEMENTS	viii
LIST OF TABLES	xii
LIST OF FIGURES	xiv
I INTRODUCTION	1
II LITERATURE REVIEW	6
2.1 Lean, Agile and Kanban	6
2.2 Queueing Models	10
2.3 Simulation in Software Development Teams	12
2.4 Clearing Functions	15
2.5 Machine Learning	17
2.6 Multi-Objective Optimization	20
III PROBLEM DESCRIPTION	23
IV SOLUTION APPROACH AND THE SIMULATION MODEL	29
4.1 Simulation Model	30
4.2 Initial Model	31
4.2.1 Stations	32
4.2.2 Work Item Types	32
4.2.3 Capabilities	33
4.2.4 WIP Limits	35
4.2.5 Blocking Scenario	35
4.2.6 Base Parameters	37
4.3 Extended Model	39

4.3.1	Multi-Capability Resources	39
4.3.2	Work Item Type Priorities	40
V	INITIAL SOLUTION MODEL	42
5.1	Multi-Capability Resource Optimization Model	43
5.1.1	Parameter Values	45
5.2	Greedy Heuristic Model	48
5.3	Two-Step Model	49
5.4	Decision Tree Clearing Function	51
5.4.1	Decision Tree Regression	52
5.4.2	Decision Tree Integrated Overall Mathematical Model	53
5.5	Numerical Analysis	55
5.5.1	Heuristic Results	55
5.5.2	Two-Step Model Results	58
5.5.3	Decision Tree Clearing Function Results	59
5.5.4	Sensitivity Analysis	61
5.5.5	Findings	64
VI	EXTENDED SOLUTION MODEL	67
6.1	Capability-Integrated Simulation Model	67
6.1.1	The Number of Possible Simulations	68
6.1.2	The Resource Optimization Model	71
6.2	Extended Solution Model	71
6.3	Machine Learning Models	72
6.3.1	Predictions up to 10 Resources	75
6.3.2	Exploring Resources up to 15	77
6.4	Insights	79
VII	MULTI-OBJECTIVE OPTIMIZATION	81
7.1	Multi-Objective Optimization Approach	81
7.1.1	Formulation of Multi-Objective Function	82

7.2	Multi-Objective Evolutionary Algorithms	83
7.2.1	Strength Pareto Evolutionary Algorithm-II	84
7.2.2	Pareto Archived Evolution Strategy	84
7.2.3	Non-dominated Sorting Genetic Algorithm-II	85
7.2.4	Numerical Analysis	85
7.2.5	Decision Making	90
7.3	Discussion	93
VII	CONCLUSION	94
	REFERENCES	98
	VITA	105

LIST OF TABLES

1	Analysis, Development, and Test Activities	25
2	Work Item Type - Capability Matrix	34
3	Processing and Block Times as Parameters	37
4	Block Percentages	38
5	t_{ik} ; capability of resource to handle work item types	46
6	c_{jk} ; cost of each resource capability	47
7	Output Values	48
8	Steps of the Heuristic Model	56
9	Resource Setting of the Final Result of the Heuristic Model	57
11	Resource Setting of the Best Result of the Simulations	58
10	Heuristic Model - Resource Allocation	58
12	Decision Tree Clearing Function Model Resource Allocation	60
13	Sensitivity Analysis Value and Cost Sets	61
14	Sensitivity Analysis	62
15	Sensitivity Analysis - Profit Scenarios	63
16	Profit Scenarios - Additional Decision Tree Depths	64
17	Best Performing Method Based on Scenario Types	66
18	Preferred Solutions	69
19	Preferred Solutions	70
20	Machine Learning Model Comparison	75
21	Machine Learning Predictions	76
22	Prediction and Simulation Results up to 15 Resources	78
23	XGBoost Performance at Output and Lead Time	83
24	Parameters for SPEA-II, PAES and NSGA-II	86
25	Aggregated Results of MOEA	87
26	Impact of Mutation Probability on MOEAs	89
27	Impact of Number of Generations on MOEAs	90

28	Final Results of MOO	91
29	Feature Set of Final Choice	92



LIST OF FIGURES

1	A Team Applying Kanban Method	2
2	M/M/c/k Queue	11
3	Examples of Clearing Functions [1]	16
4	Solution Approach	30
5	Kanban Board with 3 Work Item Types	33
6	Blocking Scenario	36
7	Initial Solution Model	42
8	Diagram of the Heuristic Model [2]	49
9	Diagram of Two-Step Model [2]	50
10	Diagram of Decision Tree Clearing Function Model [2]	52
11	Decision Tree Regression Fit on the Simulation Results Max Depth: 5 [2]	53
12	Decision Tree Regression Final 3 Nodes at Max Depth: 5 [2]	60
13	Initial vs Extended Simulation Model	68
14	Extended Solution Model	72
15	Multi-Objective Optimization Approach	82
16	SPEA-II	84
17	PAES	85
18	NSGA-II	86
19	3D Scatter Diagram of Aggregated MOEAs	88
20	Pareto Front of Final Results and Final Choice	92

CHAPTER I

INTRODUCTION

In today's world, every organization relies on information and communication technologies (IT). IT capabilities play a crucial role in determining the future trajectory of an organization, while those failing to keep up risk jeopardizing their very existence. Marc Andreessen's words "software is eating the world" are even more accurate considering the recent advancements in machine learning (ML), and artificial intelligence [3]. In addition to large technology companies; banks, automotive, insurance and energy industries employ tens of thousands of IT professionals. In such a crucial field for organizations, it is critical to find the best way of working and the optimum resource allocation. Effective resource utilization not only maximizes throughput and minimizes cycle time but also helps to control costs and increase the overall efficiency of the system. Agile ways of working [4] provided a sustaining solution for the world of software development, offering iterative production in short cycles, focused on the value delivered to the customer via small self-organized teams [5]. Among popular agile frameworks, Kanban with its roots in lean manufacturing and project management, has shown significant efficacy in managing software development schedules [6]. The Kanban method [7], which emphasizes continuous improvement, visual management, and work in progress (WIP) limits, is becoming increasingly popular as a way to manage workflow and optimize resource utilization. Kanban provides a pull-based approach that helps teams prioritize work, reduce bottlenecks, and improve overall workflow efficiency. However, optimizing resource allocation in a Kanban system can be challenging due to the dynamic and complex nature of the system. This thesis aims to address this challenge by presenting a holistic solution approach to resource

optimization of IT teams using Kanban.

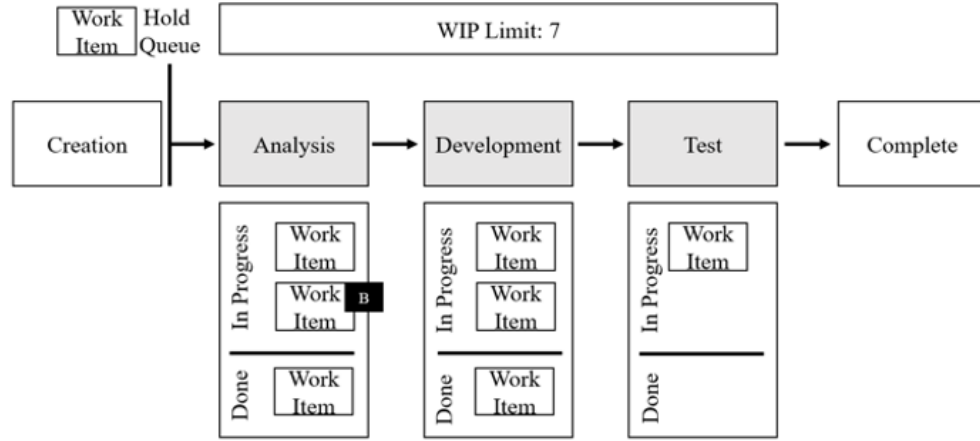


Figure 1: A Team Applying Kanban Method

A sample use case of the Kanban method applied by a team delivering software products is explained in Figure 1. In this figure, the work is visualized in a Kanban board which has three stations: analysis, development and test. At each station, work items are clustered into “In Progress” and “Done” groups. Within these stations, the “In Progress” section represents when work is being processed, and once the work is complete for the station, work items are moved to “Done” section which represents the work items that are processed, and are waiting to be pulled by the next station. If the resources in the next station can start processing a new item, they will pull one, otherwise the work will wait until they can.

Moreover, Figure 1 also embodies a WIP limit written on top of analysis, development, and test stations. WIP limit is the principle of limiting the total number of work items being processed at any given time. According to the Kanban board in the figure, the sum of all work items in analysis, development, and test stations is equal to the total work in progress, and this sum can not be more than the WIP limit of seven. The sample representation in the figure shows that there are three items in

analysis, three items in development, and one item in test, which consumes all WIP limit. Thus, the next item is waiting at the "Hold Queue" until one work item is completed.

Figure 1 also demonstrates the novel blocking scenario of this study. One of the work items in analysis station has an indicator "B" which shows that the item is blocked. This blocking scenario demonstrates that the person working on the work item is stuck for a reason, and not able to proceed. One typical reason for such a block is a dependency on other parts of the organization, which requires waiting for information, another work process, and solving a problem. The resource working on this item, may mark it as blocked, and pull another work item as long as the WIP limit allows, since the blocked items are still part of the total work in process. Thus, applying WIP limit forces teams to prioritize blocks and focus on the work in progress before starting new ones. In typical machine breakdown scenarios, the process halts until the machine is fixed. From this perspective, the blocking mechanism of teams working with Kanban is different from manufacturing.

In this study, a Kanban system simulation is designed with a blocking mechanism, varying levels of resources, WIP limits, work types, processing stations, and priorities. The simulation design provides a baseline for analyzing the performance of different scenarios. The goal of the study is to find the best strategies to maximize output with minimum cost by choosing the best parameter set that includes resource allocation, WIP limits, and priorities. The rest of the thesis is organized in the following way:

Chapter 2 provides a literature review on lean, agile and Kanban, simulations in IT and software development teams, queueing models, clearing functions, machine learning, and multi-objective optimization (MOO). The literature review presents a comprehensive overview of the origins of lean, the evolution of agile and the Kanban method, key principles of Kanban, and its applications in software development and IT. It also reviews the current state of research on resource optimization in Kanban

systems and the decision analysis methods used in the field. Then, queueing models focusing on Kanban and IT teams are investigated, followed by resource optimization. Finally, an overview of clearing functions, machine learning methods and robust optimization are covered. The literature review provides a theoretical foundation for the rest of the thesis and identifies the research gaps that this study seeks to address.

Chapter 3 describes the problem statement, details of the Kanban system and presents an overview of the problem, the complexity of the team dynamics and why it is impractical to model it using queueing theory. It explains the blocking scenario in detail with practical examples. The chapter also provides insights regarding how the organizations are dealing with the current problem.

Chapter 4 presents the proposed solution framework and its implementation in the Kanban system simulation. The chapter starts with the holistic solution framework and depicts the methods applied in this study. Then, the simulation model, software, hardware, and used libraries are described. The chapter explains how comprehensive simulations including various parameter combinations serve to iterate in the heuristic model and to gather data for analysis in the other solution methods.

The simulations are followed by an in-depth explanation of the initial solution approach in Chapter 5. The initial solution includes greedy heuristic model, a two-step model, and a decision tree clearing function and their underlying principles, assumptions, and limitations. Furthermore, the sensitivity analysis is provided to compare the models and find the differences, advantages, and disadvantages.

Chapter 6 extends the initial solution model by integrating resource capabilities into simulations. This solution model uses a new simulation set to collect data, and studies machine learning models on this data to investigate best strategies to find the optimum resource distribution, WIP limit, and priorities.

Chapter 7 takes a holistic view of the complexity of Kanban through MOO. The principles of Kanban include minimizing production time and maximizing output

efficiently by keeping costs at a minimum. MOO and a Pareto analysis enables an overview of the system from these different perspectives.

Chapter 8 concludes the thesis and provides recommendations for future research. The chapter summarizes the key findings of the study, including the effectiveness of the different solution approaches and the factors that impact resource optimization in a Kanban system. It also discusses the implications of the research for software development and IT teams and provides recommendations for future research on resource optimization in Kanban systems.

Overall, this thesis contributes to the field of decision analysis in IT and software development teams' resource planning by presenting a comprehensive approach to resource optimization in Kanban systems. The simulation model and the solution approaches presented in this study provide a framework for evaluating and optimizing resource allocation in a dynamic and complex system. This approach can help software development and IT teams improve their workflow efficiency, reduce costs, and increase the overall effectiveness of their operations. The findings of this study also have implications for decision makers in other industries that use Kanban or similar methods for managing workflow and resource allocation.

CHAPTER II

LITERATURE REVIEW

This study addresses a resource optimization problem of a software development team working with Kanban by utilizing simulations, heuristics, clearing functions and machine learning techniques. The literature review begins by providing an overview of related fields, including lean, agile, and Kanban, and explores simulations in software development teams. The following section investigates queueing models, since the Kanban system is a complex queueing system. Due to the difficulty in finding closed-form solutions to queueing systems, approximations or simulation tools are commonly employed for modelling and solution generation. The literature review then examines clearing functions, which help to capture the relationship between characteristics, the current state of the manufacturing system, and desired outputs, such as cycle time or throughput. Then, an overview of machine learning methods is provided, and finally, the literature on multi-objective optimization, is covered.

2.1 Lean, Agile and Kanban

Lean manufacturing is a management philosophy and approach that aims to improve the efficiency and effectiveness of production processes by eliminating waste, reducing lead time, and improving quality. Originally developed and implemented by Taichi Ohno and Shigeo Shingo [8] at the Toyota Motor Company in Japan in 1940s, lean manufacturing is characterized by its focus on continuous improvement, respect for people, and the pursuit of perfection.

The primary goal of lean manufacturing is to maximize customer value while minimizing waste. This is achieved by identifying and eliminating non-value-adding

activities (or "waste") in the production process, such as overproduction, waiting, defects, excess inventory, overprocessing, unnecessary motion, and underutilized talent. By eliminating waste, the production process becomes more efficient, productive, and profitable, and ultimately leads to better quality products and services delivered to customers in a timely manner.

Lean manufacturing involves a set of principles and tools, such as value stream mapping, 5S, Kanban, Just-in-Time (JIT), Kaizen, and Poka-yoke, that help organizations to identify and eliminate waste and continuously improve their processes [9]. The implementation of lean manufacturing requires a cultural shift towards continuous improvement, teamwork, and a focus on customer value, and involves the active participation and engagement of employees at all levels of the organization.

One of the tools of lean manufacturing is Kanban, which means "signal" or "card" in Japanese [10]. Kanban is a visual control system that helps to manage and control the flow of work, inventory, and resources in a production process. In Kanban, each stage of the production process is represented by a card or a signal, which indicates what work needs to be done, how much of it needs to be done, and when it needs to be done. When a stage has completed its work, it signals the next stage to begin its work by passing on the card or signal. This ensures that production flows smoothly and efficiently, without any interruptions or delays.

Kanban is designed to help organizations achieve a "pull" system of production, where work is only produced when there is demand for it, rather than a "push" system, where work is produced based on a forecast or a schedule. This helps to reduce waste and improve efficiency by minimizing overproduction, excess inventory, and waiting time. Kanban can be used in various industries and processes beyond manufacturing, including software development, healthcare, and service industries. In these settings, Kanban is used to visualize and manage the flow of work, to identify and eliminate bottlenecks and waste, and to continuously improve the process.

Lean philosophy not only inspired manufacturing, and management, but also project management, and software development. Projects in IT industry were being handled in strictly separated phases by individuals with different skills taking different roles. Gantt charts (1903) [11], and Critical Path Method (1957) [12] were still the main methods of project management until 1990s. In 1986, Takeuchi and Nonaka [13] compared different ways of project management and used the word "scrum" to describe an overlapping product development approach that removed the boundaries between phases.

In the 1990s, agile methodologies began to emerge in response to the limitations of traditional project management approaches, which were seen as rigid and inflexible. In 1995, Ken Schwaber and Jeff Sutherland explained Scrum as a way of working for software development teams [14]. The agile manifesto (2001) represents a key turning point in IT and software development industry [4]. It was heavily influenced by lean philosophy and consisted of four values and twelve principles with the focus of collaboration, adaptability, value, and continuous improvement.

The Agile movement continued to evolve in the 2000s, with further development of Agile ways of working, such as Scrum and Kanban. Scrum became a framework for self-organized teams in software development [15] and gained worldwide recognition, followed by introduction of various scaling frameworks that address organizational level agile adoption [16]. Similarly, Kanban was adapted to IT industry by David Anderson [7]. In 2010, David Anderson published his implementation of Kanban in the IT industry, based on the original principles of lean manufacturing. His approach to Kanban involves several key steps. First, he recommends visualizing the workflow by creating a Kanban board, which is a visual representation of the work that needs to be done, and the stages that the work must go through to be completed. This helps to identify bottlenecks, reduce waiting time, and improve flow. Next, Anderson recommends implementing a pull system, where work is only produced when there is

demand for it, rather than a push system, where work is produced based on a forecast or a schedule. This helps to reduce waste and improve efficiency by minimizing overproduction, excess inventory, and waiting time. Anderson also emphasizes the importance of limiting WIP, which means that only a certain number of tasks can be in progress at any given time. This helps to improve flow and reduce waiting time, as well as to prevent team members from becoming overwhelmed with too much work. Finally, Anderson stresses the importance of continuous improvement, which means that teams should constantly be looking for ways to improve their processes and workflows. This involves analysing data, identifying areas for improvement, and implementing changes to improve flow and reduce waste.

The Kanban Method has five properties:

1. Visualize the Workflow
2. Limit WIP
3. Measure and Manage Flow
4. Make Process Policies Explicit
5. Use Models to Recognize Improvement Opportunities

Overall, the development of lean philosophy and agile methodologies has been a continuous process that has been shaped by the changing needs and demands of the software development industry. While lean philosophy laid the foundation for agile methodologies by emphasizing the importance of minimizing waste and maximizing customer value, agile methodologies built on this foundation by emphasizing flexibility, collaboration, and iterative development.

The ongoing evolution of lean and agile frameworks reflects the continuing efforts of software development professionals to improve their processes and deliver better value to their customers. According to 16th State of Agile report [17], Scrum is leveraged by 89% of participants whereas Kanban is used by 56% some of which is a combination of both. Generally, Scrum is chosen for adaptive, incremental delivery,

and Kanban for predictable, effective flow of work. In 2017, Lei showed through a statistical study that Kanban was more effective in terms of managing project schedule and achieving deadlines [6].

2.2 Queueing Models

A queueing model is a system that processes flowing items in need of a service. The items wait for service due to limited capacity and when their service is complete, they immediately leave the system [18]. One of the pioneering and the most significant research dates back to 1909 from A.K. Erlang about queues in telephone lines [19]. Since then, queueing models have been studied as a branch of applied probability theory and became a solid research area with strong foundations. The models are generally represented using Kendall's notation [20] via letters and numbers in parentheses which reflect firstly arrival, secondly service, thirdly capacity and lastly queue size. Gaver introduced blocking as random interruptions at a single-server system with priorities [21]. A groundbreaking research in the field is the jobshop-like queueing systems which proposed a multi-server production line with priorities [22]. Since 1960s, various models have been proposed, the most common queueing models including "Birth-Death (M/M/1)" processes, "Multi-Server" systems, "Finite Capacity (M/M/1/K)" queue, "Multi-Server Finite Capacity (M/M/c/K)" queue, "Finite Source (M/M/c/M)" systems, "State-Dependent" service, "M/G/1" queue, and "G/M/1" queue [23]. Figure 2 presents a representative "M/M/c/K" queueing system.

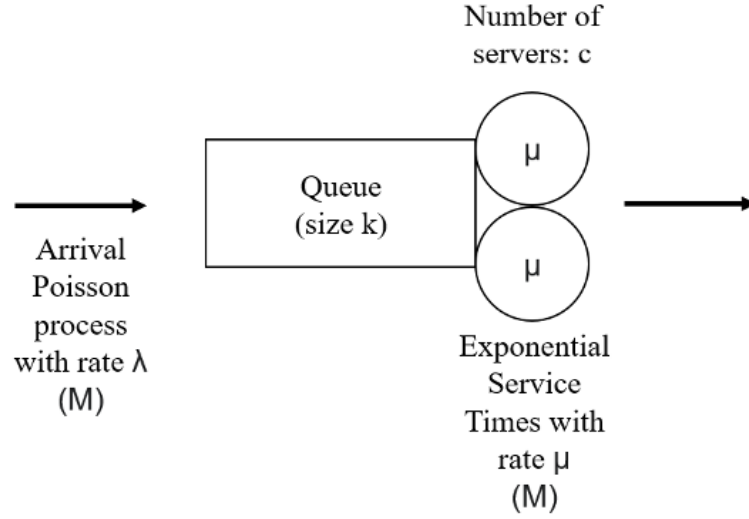


Figure 2: M/M/c/k Queue

Since Kanban is a method which represents the flow of value, it also is a queueing system. Several researchers have proposed queueing models to analyse Kanban systems. Buzacott proposed a queueing network that consists of series of multiple server queues with a blocking mechanism which is represented as an M/M/C queue [24]. Buzacott's model also includes a blocking mechanism, which occurs when a server in the queue is idle but cannot start processing the next item because the corresponding Kanban card is not available. This can happen when a station is not yet ready to receive the next item, or the next item cannot be brought to the station. Using these assumptions, Buzacott derives formulas for various performance metrics, including the mean waiting time and the mean number of items in the system. The model can be used to analyse the impact of different parameters on the performance of the Kanban system, such as the number of servers, the arrival rate of items, and the rate of Kanban card flow.

Berkley provides a review of literature modelling Kanban systems [25]. In his review, he describes blocking by total queue size, and blocking by parts, and derives calculation of key metrics like waiting time, lead time, and effects of queue size.

Boucherie et al. define a Kanban system as an arrival first system and describe two different blocking scenarios [26]. The first scenario is the blockage of the production line when a station does not have enough items, and wait for the items to arrive, whereas the second scenario uses an emergency spare part to proceed.

Based on literature related to queueing models, it is possible to state that the Kanban System subject to this dissertation is a combination of various models, and may be called a multiserver priority queueing network with feedback. In addition, the specific system includes a blocking that differentiates with the previously addressed blocking methods. In the case of this study, the interruption happens to the work item and the server (the resource/ agent) keeps the capability of working on another item. The reason of this difference is the manufacturing focus of the previous researches, and the distinct nature of IT and software development.

Eventually, Kanban systems have been subject to thorough research, and have been mathematically modelled as queueing models, but there is no queueing model in the literature describing the Kanban system of a software development team subject to this study.

2.3 Simulation in Software Development Teams

Due to the complexity of the system to be modelled, the use of simulations for resource optimization in software development has gained interest of researchers. Using simulations to model various aspects of the software development process (including task allocation, scheduling, and resource utilization) in order to optimize resource allocation and improve project outcomes is called Software Process Simulation Modelling (SPSM) [27]. SPSM has been attracting researchers, especially in agent-based simulations and discrete-event simulations [28]. This literature review provides an overview of the current state of research in this area.

Glass and Vessey [29] provide an early example of the use of simulation in software development research, and they highlight the importance of understanding the nature of software development work in order to optimize the development process. Barghouti and Rosenblum [30] apply simulation to conduct an analysis for a software maintenance process.

After 2000, with the rise of agile ways of working, several researchers have studied effects of agile and Kanban to software development teams. In their paper, Melis et al. simulate software development process and the impact of extreme programming on productivity [31]. The paper uses simulations to show test-first programming and pair programming lead to significant improvement in software quality and productivity. Similarly, another research by Turnu et al. investigates benefits of applying agile practices in software development using simulation [32]. Their findings also show that agile practices can increase software quality, productivity, communication, and collaboration. Anderson et al. [33] developed in 2011 an event-driven simulator for a team working with Kanban focusing on software maintenance, whereas in 2012 used simulations to compare Kanban and Scrum teams [34]. A further study comparing the effects of Scrum and Kanban on software development projects was conducted in 2017 [6]. Authors found in their statistical analysis that Kanban is better at managing project schedules.

Simulations also enabled researchers to investigate resource allocation, project scheduling, and lead times. Otero et al. [35] propose a systematic approach for resource allocation in software projects. The approach involves using mathematical programming to optimize the allocation of resources to tasks based on the project requirements and constraints. The proposed approach aims to balance the workload of resources, minimize the project duration, and maximize the project quality. The authors present a case study to demonstrate the effectiveness of the proposed approach in a software development project. The results show that the proposed approach

can provide a significant improvement in resource allocation compared to traditional manual methods.

Tregubov and Lane incorporate prioritization to a scheduling algorithm which simulates Kanban systems while comparing different ways of prioritization [36]. One of the ways of prioritization used in this study is the value of each work item with the goal of gaining highest value at the end of the simulations.

Wysocki approaches from another perspective, using agent based simulations for project management in combination with object-oriented models to estimate project completion dates for large projects following agile ways of working [37].

Weffen et al.'s work on Kanban [38] estimates lead time using a probabilistic model based on influence diagrams. The goal of the study is to help prioritization and improve efficiency of the team by providing estimates and automating some of the estimation and planning efforts.

Due the complexity of modelling the flow in Kanban, researchers used simulation techniques to see the results by providing parameters. An event-driven simulation was conducted by Anderson et al. [11] which took inputs of resources, work completion times, and provided results including lead time, and amount of work items completed.

Damij and Damij conducted another study based on simulating the flow of a Kanban team, and improving the output [39]. Their focus was on the selection of resources and finding the right change for maximum output increase. Their study may be considered a simplistic version of the greedy heuristic method applied in this study.

Examples of combining simulation with linear programming are followed by a well-known incorporation of an iterative method proposed by Hung and Leachman to incorporate simulations and linear programming for automated production planning [40].

Simulation is a powerful and effective tool to analyze software development teams.

Researchers have shown increasing interest in simulating software development teams working with Kanban in the last two decades which enabled models to prioritize, estimate lead time, and show the benefits of using Kanban. On the other hand, none of these simulations modelled blocking and combined this type of Kanban with clearing functions or machine learning methods. From these perspectives, this study proposes a key contribution to its area.

2.4 Clearing Functions

The complexity of modelling manufacturing systems using queueing theory and stochastic methods, led the researchers to explore new ways. Clearing functions have emerged from this need by forming the relationship between resources, throughput, WIP or lead time implicitly. Srinivasan (1988) proposed a model that optimizes the allocation of manufacturing resources based on the clearing functions that relate the WIP inventory levels to the production and scheduling decisions [41]. The model is based on a Lagrangian relaxation technique and is shown to be effective in minimizing the total production costs and improving the system performance.

Clearing functions have been used in various forms to address and capture the complex relationship between throughput, utilization, workloads, WIP, and lead times in manufacturing systems. Graves defined a linear function [42] whereas Srinivasan et al. [41] and Karmarkar [1] proposed nonlinear functions as clearing functions.

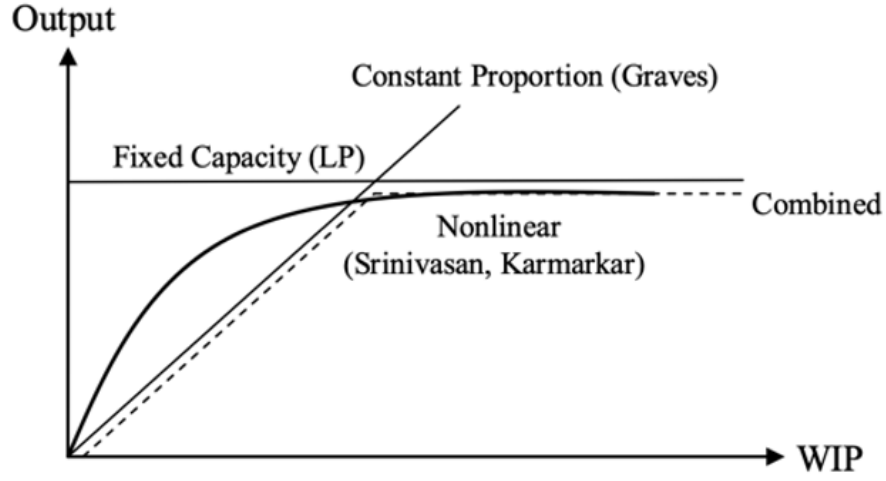


Figure 3: Examples of Clearing Functions [1]

The popularity of clearing functions grew in time with further research and applications. Asmundsson, Rardin, and Uzsoy (2002) introduced tractable nonlinear capacity models for production planning, which used non-linear clearing functions to determine how much work could be released to each machine given the capacity, production, and inventory constraints [43]. The authors proposed a linearization of the non-linear clearing functions and showed that the use of clearing functions improved the efficiency of the production process and reduced costs.

Pahl et al. presented a production planning model that considered load dependent lead times, which used clearing functions to determine how much work should be released to each machine [44]. The clearing functions used in this model were able to model the manufacturing systems in a higher accuracy in comparison with the conventional models.

Selcuk et al. [45], Missbauer et al. [46] both used clearing functions to determine the optimal WIP levels. Selcuk et al. used a discrete event simulation model to represent the supply chain, and they used clearing functions to control the flow of WIP

inventory through the system. They compared different policies based on their impact on various performance measures, such as cycle time and inventory levels. Missbauer et al. showed that the use of clearing functions led to significant improvements in performance and reduced costs. Albey and Uzsoy used clearing functions to estimate lead times in addition to determining the optimal levels of WIP [47].

Albey et al. conducted an exploratory study of disaggregated clearing functions for production systems with multiple products [48]. The authors used a simulation-based approach to evaluate the performance of their proposed method and compare it with the traditional aggregated method. The study showed that the use of disaggregated clearing functions improved the accuracy of production planning and reduced costs.

In 2009, Irdem evaluated the fitting methodology and performance of clearing functions in production planning models [49]. The study shows that the choice of fitting methodology had a significant impact on the performance of clearing functions and that the use of more complex functions did not necessarily lead to better performance.

Overall, the use of clearing functions in production planning has been shown to lead to significant improvements in performance and reductions in costs. Clearing functions have been applied in a wide range of production planning models and have been found to be effective in improving machine utilization, reducing lead times, and reducing inventory levels.

2.5 Machine Learning

One of the most established methods of analyzing data generated by the simulations is machine learning. The term machine learning was first coined by A.L. Samuel in 1959 [50] as a way of programming a computer to learn playing a game given rules, a sense of direction, and playing time. Today, the term machine learning refers to the automated detection of meaningful patterns in data without providing a direct formula

[51]. With the advance of computation technology, various ML algorithms have been developed and successfully applied to tackle problems including text classification, natural language processing, speech processing, vision applications, computational biology, fraud detection, automated vehicle control [52]. Machine learning has been applied in optimizing resource allocation, and resource scheduling problems in servers, network, and cloud computing [53], [54], [55]. Commonality of applications include an accurate prediction in complex environments in which mathematical modelling is not possible. The application of machine learning in optimizing resource allocation in software development projects is a relatively new area of research. The study by Pospieszny et al. [56] demonstrate an application of machine learning algorithms in effort and duration estimation, without combining with simulations. Song et al. [57] provide a review on the interactions between machine learning and optimization which outlines areas where machine learning algorithms are used to improve optimization results. Veeranjanyulu et al. demonstrate the effectiveness and accuracy of machine learning algorithms in combination with simulations to predict and help optimise air quality [58]. Some of the most used machine learning techniques include random forest, XGBoost, and neural networks. Each model offers unique strengths in handling large and complex datasets, making them well-suited for analyzing the multifaceted dynamics of Kanban resource allocation. random forest is an ensemble learning method, particularly effective for regression tasks [59]. It operates by constructing a multitude of decision trees during training and outputting the mean prediction of the individual trees. This method is known for its high accuracy, ability to run in parallel, and robustness to overfitting, primarily due to its ensemble nature. Random forest regression can handle a large number of features and is adept at capturing non-linear relationships, making it an excellent choice for complex datasets like those found in software project management. The second ML algorithm considered in this study is XGBoost. Boosting refers to a class of learning algorithms that fit

models by combining many simpler models [60]. These simpler models are typically referred to as base models and are learnt using a base learner or weak learner. These simpler models tend to have limited predictive ability, but when selected carefully using a boosting algorithm, they form a relatively more accurate model. This is sometimes referred to as an ensemble model as it can be viewed as an ensemble of base models. Friedman [61] developed gradient boosting and showed that it can be interpreted as a gradient descent algorithm in function space. XGBoost (Extreme Gradient Boosting) is another ensemble technique, renowned for its efficiency and performance which is based on gradient boosting [62]. It is an implementation of gradient boosted decision trees designed for speed and performance. XGBoost is known for winning machine learning competitions by its scalability and effectiveness in the accuracy of predictions [63]. The model's ability to handle sparse data and its efficient computation of gradient statistics make it highly suitable for large-scale and complex datasets. Neural networks (artificial neural networks, or in short NN), inspired by the structure and function of the human brain, are a set of algorithms modeled as human neurons [51]. First implemented by Rosenblatt [64], NN consists of a large number of basic computing devices (neurons) that are connected to each other in a complex communication network, through which the brain is able to carry out highly complex computations. NNs are particularly good at recognizing underlying relationships in data through a process that mimics the way the human brain operates. In regression tasks, NNs excel in their ability to model complex, non-linear relationships. Their deep learning capabilities allow them to process large volumes of data with many layers of abstraction, making them exceptionally powerful for modelling the complicated dynamics in Kanban-based environments. The versatility and adaptability of NNs make them an ideal choice for tackling the intricate patterns and relationships in resource allocation and workflow management.

2.6 *Multi-Objective Optimization*

Software development teams working with Kanban take many factors into account and aim to deliver the maximum output within the minimum possible time at the highest quality. As a whole, this is a complex task. Solving optimization problems for more than one goal is described as multi-objective optimization (MOO) [65]. MOO problems exist in various fields of nature, and civilization. For this reason, the field has attracted researchers' attention, and extensive research has been conducted, and methods have been developed for decades [66].

Foundations of MOO and decision-making under trade-offs can be dated back to 1896 to Pareto's work in economics [67]. Considering MOO and any two feasible solutions x and y , x dominates y if x is not inferior to y on all objectives and x is superior to y on at least one objective. If x is not dominated by all the others in the solution set, then x is part of the trade-off optimal solution, also called the Pareto front or Pareto optimal solution set [68]. This solution set consists of elements that cannot be improved without sacrificing one of the objectives. The Pareto front favours one of the objectives more than the others at different levels, and decisions may be made according to the importance of each objective. Deb [69] suggests taking a two-step approach in which the first step consists of finding the trade-off points whereas the second step consists of taking potential other non-quantitative information that was not represented before into account and making the decision.

Researchers have primarily employed two main groups of methods to solve MOO problems; scalarization and evolutionary algorithms [70]. The classical scalarization methods include reformulating objective functions into constraints of a single objective optimization problem and solving the single objective problem. A common scalarization technique is linear weighting. In linear weighting, weights are assigned to the objective functions. This technique faces the challenge of the determination of weights.

The second group; evolutionary algorithms offer an effective potential solution to this challenging MOO problem via their population and evolution-based nature. Early research published in the 1990s, in multi-objective evolutionary algorithms (MOEA), an initial population of solutions is created which then evolves over successive generations by selecting the fittest solutions according to the objectives and applying genetic operators to generate next populations [71], [72]. The process stops when one of the defined criteria is met, or a certain number of iterations are conducted. Using MOEA, through a single run, a strong solution set can be reached [73].

Significant MOEAs demonstrated effective solutions with features regarding ranking, splitting the population, generating new offsprings, creating diversity, and archiving. Fonseca and Fleming [74] introduce a rank-based fitness assignment method where ranks are assigned according to domination and fitness scores are assigned comparing the ranks. Strength Pareto evolutionary algorithms (SPEA and its improved version SPEA-II) use strength at individuals in the population as a type of fitness which represents it's dominance against other solutions [75]. The algorithm incorporates density estimation to maintain diversity, uses crossover and mutation to create generations. Pareto archived evolution strategy (PAES) [76] is a method that starts with a random initial solution and mutates this solution to create a new generation. Comparing the new generation to the one before in terms of domination, the non-dominated generation becomes the actual population whereas the old one is checked to be archived. To maintain diversity, PAES uses an adaptive grid which is a histogram technique. Non-dominated Sorting Genetic Algorithm II (NSGA-II) [77], sorts the population into two groups based on domination. After sorting, a crowding distance is calculated to maintain diversity for the crossover and mutation to produce an offspring population. After certain generations, the process terminates.

Using MOEA, the decision maker does not face the challenge of assigning weights to different objectives. On the other hand, the decision maker faces a set of points to

be assessed. Based on the preferences within the set of points, evolution is conducted and a new generation of points is created. MOEA provides an effective and promising solution for challenging multi-objective optimization problems.

As a system, Kanban aims to minimize lead time and maximize output. Thus, the nature of Kanban is a direct field of application for MOO studies. Hence, research bringing Kanban and MOO has been limited. Baradaran's work combines a Kanban system and simulations against the goals of lead time and output while staying at a simplistic level of linear optimization and comparing the simulation results [78]. Another study focuses on a manufacturing application of Kanban, and uses a genetic algorithm supported with simulations to solve the MOO with objectives of production cost, and transportation cost [79]. Prasad et al.'s work adopts a genetic algorithm for multi-objective scheduling of a Kanban system that has the objectives of minimization of mean completion time and standard deviation of mean completion time [80].

The application of MOO in Kanban systems aligns closely with the built-in goals of maximizing output while minimizing lead time and costs. The evolution of MOO approaches, particularly in evolutionary algorithms, offers new avenues for optimizing Kanban processes.

In summary, the literature extensively covers agile, Kanban, optimization, simulation, and machine learning fields. Hence, resource optimization of software development teams using Kanban is underexplored. This study aims to fill this gap by applying simulations, and machine learning algorithms to optimize resource distribution in a software development team using Kanban. The integration of these diverse fields presents a novel approach, promising insights into more efficient and effective decision making in the rapidly evolving landscape of software development. erinc

CHAPTER III

PROBLEM DESCRIPTION

This chapter aims to provide a detailed description of the problem addressed in this study, along with its complexity and the practical effects of the existing situation. The chapter begins by introducing the team working with Kanban, their roles, their work, and data.

The roots of software development can be traced back to the 1940s [81]. Initially, programmers wrote code by hand on paper. However, with the development of computers in the 1950s and 1960s, much of the coding process became automated, and programming languages like FORTRAN and COBOL were introduced. As software development became more complex in the 1970s and 1980s, the need for better tools and methodologies arose, leading to the creation of structured programming and software development life cycle (SDLC) models like the Waterfall model [82].

The software development and software development lifecycle industries grew rapidly in the 1980s, leading to the emergence of new IT organizations and software companies that offered software products and services. This also resulted in the rise of cost-effective offshore IT service providers. This dynamic gave rise to the business analysis role [83], where professionals understood the business needs and communicated them to the developers in a technical manner. They were also responsible for accepting the built product and confirming its implementation. Since then, the business analysis role has become a foundational component of software development teams, along with software developers and testers.

The software development team subject to this study is part of a unit that delivers enterprise applications for a telecommunication organization. The team consist of

three roles; business analysts, software developers, and testers. The business analysts are responsible for understanding the business needs and translating them into technical requirements that can be used by software developers. The analysts work with internal stakeholders to identify and analyse business problems, and then propose solutions to those problems. They act as a bridge between the business side of the organization and the technical side, ensuring that the software being developed meets the needs of the business. Software developers are responsible for designing, coding, and maintaining software systems. They use programming languages and other tools to create applications that solve business problems or meet user needs. They work with the technical requirements provided by the business analysts and create software solutions that are efficient, scalable, and maintainable. In this specific team, software developers design and maintain databases, web services, workflow applications, user interfaces. They also collaborate with testers to ensure that the software is free of bugs and meets the quality standards. Testers are responsible for ensuring that the software developed by the software developers meets the business and technical requirements provided by the business analysts. They test the software to identify and report any bugs, errors, or defects, and then work with the software developers to fix them. They use a variety of testing methods, including manual and automated testing, to ensure that the software is of high quality and meets the needs of the business. They also work closely with business analysts and software developers to ensure that the software meets the specifications and requirements of the stakeholders. Finally, business analysts, developers, and testers share the responsibility of shipping the code to the live environment using continuous deployment techniques.

The team has adopted the Kanban methodology as their preferred way of working due to its ease of implementation through the "Start with what you have" principle [7]. Kanban, as proposed by David Anderson, is an evolutionary approach that provides flexibility in the team's roles and processes. In contrast, Scrum is a more

revolutionary approach that requires significant changes to the team’s roles and does not explicitly include business analysts or testers. The team’s Kanban board visualizes their workflow, which consists of three stations: analysis, development, and test. Work items are kept in a backlog until an analyst pulls them into the analysis station, where they are analyzed according to stakeholder needs. Business analysis is typically documented in an analysis report, which outlines the business requirements, change impacts, possible scenarios, and error messages. Once the analysis is complete, a developer pulls the work item for software design, coding, and implementation. The software undergoes an initial test by the developers to confirm that it meets the business scenarios, often with the aid of automated testing. The work then moves to the test station, where the tester checks the possible scenarios and runs various tests with the business logic. Any bugs found at this stage are given high priority and addressed by the developers. Activities conducted at each stage are summarized in table 1.

Table 1: Analysis, Development, and Test Activities

Table 1: Analysis, Development, and Test Activities

Analysis	Development	Test
Capturing requirements	Technical specifications	Writing test scenarios
Planning delivery	Designing the software	Conducting tests
Analyzing processes	Developing	Identifying bugs
Impact analysis	Running initial tests	Resolving issues
Analysis document	Deployment	Ensuring quality

The work that is delivered in software development is mostly unique. Although the team members gain experience and understanding of the functionality, technology, and infrastructure in time, most of the new requests are unique. Thus, it is common to end up doing research trying to find ways of delivering the business needs, and

it is not easy to estimate how much time the team needs to spend at each station. Estimating the size of tasks in IT teams is a common challenge. Story points [84] have become popular since the adoption of agile ways of working, but it is still not easy to come up with an accurate size estimate.

Although most of the work has a unique feature, the team's deliverables can be classified into three main categories: front-end, back-end, and system design. Front-end development involves designing and implementing the user interface and user experience, including aspects such as layout, appearance, and interactivity. It typically involves the use of technologies such as HTML, CSS, and JavaScript. Back-end development focuses on the server-side components, including the application's logic, database management, and integration with other systems. Programming languages like Python, Java, and PHP are commonly used in back-end development. System design, on the other hand, refers to the high-level architectural planning and decision-making process that defines the overall structure and organization of the software system. This includes identifying software components, defining their relationships and interactions, and determining the technologies and frameworks to be used. These three clusters are represented as work item types 1, 2, and 3 respectively. It should be noted that the team's analysts, developers, and testers may not have the expertise to deliver all types of work items by default. They need to work on gaining the necessary skills and experience, and many teams divide responsibilities between different work item types.

In the extended model of the solution approach, resources with capabilities of handling multiple work item types are introduced, which is a realistic scenario. The challenge faced with this feature is the decision to choose which work item to start working on. The model introduces priorities for this reason and incorporates them as parameters of the model. Each type of work is assigned a priority out of 1, 2, 3 (1 indicating the highest). Considering the blocking mechanism of Kanban, the work

items returning from block have always higher priority than items that are waiting in the queue. When multiple items of the same work type wait in the queue, the first-in-first-out principle is applied. Thus, priorities become another decision making parameter that have a direct influence on the output and lead time.

The team's work is interconnected with other applications within the organization, which can pose various dependencies. For instance, a business requirement may necessitate access to user data, which the team discovers belongs to another system within the organization. In such cases, the team has to wait for the concerned department to provide a web service for secure data exchange. Until this service is provided, the work item has to remain on hold. Other dependencies include user authentication, cloud data storage and retrieval, organizational controls, sales applications, invoicing, customer communication, and more. Due to these dependencies, the team has to contact other teams for clarification, agree on common functionality, obtain approval, and frequently wait for a fix. These dependencies are recorded as a block on the Kanban board. When an item is blocked, it waits until the dependency is resolved, which is usually done by another team. However, the resource responsible for the blocked work item may pull another work item. When the block is resolved, the work item may be pulled again at the same station with priority, rather than waiting for the entire queue.

The team employs WIP limits at the board level, as shown in Figure 1. These limits specify the maximum number of work items that can be in progress at any given time, encompassing items in the analysis, development, and test stations, including any blocked items. The WIP limit does not include items in the backlog or those that have already been completed and deployed to the live environment. Applying WIP limits encourages the team to focus on completing work items in progress. In situations where multiple items are blocked, team members may find themselves unable to pull new work items and must instead focus on finding ways of completing

the blocked items.

Due to the complex dynamics of the system, modelling and planning the work can be a challenging task. It can be difficult to determine the appropriate number of analysts, developers, and testers needed for each station, as well as the specific skills required for each team member. Moreover, deciding on the right WIP limit can also pose a challenge, as it directly affects the lead time of the work items. The blocking scenario adds another layer of unpredictability, making it hard to forecast how long it will take for each item to be completed. Given these challenges, the team is able to experiment with the WIP limits and how they affect delivery and lead time. However, there is no possibility for the team to find out how the resources and their capabilities would affect output.

In conclusion, the team faces a range of challenges, including the difficulty of estimating size, the need for different skills and experience for each work item type, and the presence of dependencies and blocking scenarios. These factors make it hard to model and plan the work, including determining appropriate resource allocation and Work in Progress limits. Despite these challenges, the team is committed to optimizing their processes and delivering high-quality work. In the next section, the proposed solution will be discussed to address these challenges and compare different ways of resource optimization with the appropriate WIP limit.

CHAPTER IV

SOLUTION APPROACH AND THE SIMULATION MODEL

The approach taken in this study employs a structured methodology to optimize the resources to maximize output and minimize lead time within a software team working with Kanban. Initially, the data obtained from a telecommunication company is used to estimate the input parameters of the simulation including processing times, blocking percentages and times. This real-life data also provides the structure for the Kanban team to be simulated including three stations and three work item types. Then, in order to prepare the simulation phase, a random input generation is conducted for the varying parameters including the resources, priorities, and the WIP limits.

The simulation structure is the novel and most complex part of the solution framework. As explained in detail in this chapter, simulations incorporate three types of work, prioritization among work items, a blocking mechanism, and multi-capability resources. Simulations are run to collect output data for further analysis, and the highest output-generating result of these simulations is kept for future comparison. Meanwhile, simulation data is used to train ML models.

The study takes a solution approach that has two different levels, combined with an exploratory section. The first level consists of the initial simulation model and its outcome data in combination with three scenarios. The first scenario is a greedy heuristic model, the second one is a two-step model and the third one is the decision tree clearing function. On the other hand, the second level extends the approach to multiple facets. To start, simulations are enhanced by integrating multi-capability

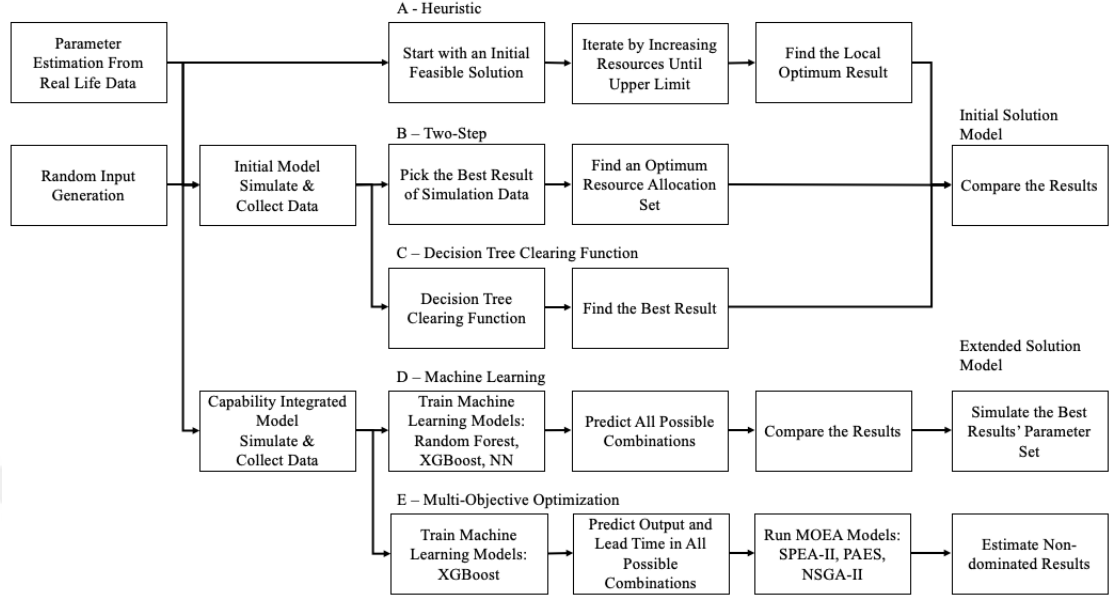


Figure 4: Solution Approach

resources and priorities. This integration significantly increases the level of complexity of the simulations and the number of possible simulation input parameters, thus making it intractable to simulate the whole parameter universe. To overcome this problem, ML algorithms are applied to investigate the limited parameter combinations to come up with effective predictions. Finally, multi-objective optimization is introduced to explore the impact of lead time together with output. The solution approach is depicted in 4 where the methods "A", "B", and "C" are part of the initial level, and "D" is part of the extended solution whereas "E" is the final exploration towards MOO.

4.1 *Simulation Model*

The Kanban System subject to this study was previously described as a "Multi-Server Priority Queueing Network with Feedback". This system is a composite of various queueing models, including multiserver queueing networks, feedback models, and priority queues. While each of these individual models has been examined in separate

probabilistic studies, integrating all of them in a single research endeavour adds a huge level of complexity. Developing a probabilistic model that accurately captures the behaviour of the system is challenging, and any such model may still deviate considerably from real-world results. Therefore, data collection from simulations and subsequent data analysis are essential in this study.

The choice of simulation software is critical for the successful construction of the model. After evaluating factors such as speed, integration, analytical capabilities, and automation potential, Python programming language was selected as the foundation for the simulations. Simulations, visualizations, and optimization models are executed end-to-end using a Jupyter notebook, which leverages the open-source discrete event simulation framework SimPy [85]. In addition, several libraries, including Numpy [86], Cplex [87], and Matplotlib [88], have been utilized throughout the study. The simulations are executed on a MacBook Pro (2015) equipped with an Intel Core i5 processor.

The process-based simulation begins with the random generation of various types of work items. Each work item is then pulled and processed by respective resources capable of handling the type of work. Once the work is complete at a station, then it waits to be pulled by the next station. Once the work at all stations is completed, the work item is done. The simulation runs for a duration of 52 weeks after a warmup period of 10 weeks. All simulation parameters are derived from data collected from a large telecommunication organization, with further adjustments made based on practical experience.

4.2 Initial Model

The initial simulation model consists of three stations, three types of work, three capabilities of resources at each station, WIP limits, and blocking scenario. The following subsections provide a detailed overview of each feature of the simulation

model.

4.2.1 Stations

The simulation consists of three main stations representing analysis, development, and test. These stations describe a typical software development value stream, which is also explained previously. In the analysis station, business requirements are captured, and an optimal solution is proposed based on the system's capabilities. The work items wait until being pulled by a respective analyst which starts the processing time. All processing times are assigned through probabilistic distributions shown in the following parameters subsection. Once analysis is complete, the work item is pulled by developers who initiate the development process. During this stage, software development is carried out based on the captured requirements. After development, the code is ready to be tested and eventually deployed to the live environment. The testing phase begins after the development process is complete. Testers carry out various tests and fix any errors that arise. The goal is to finalize the code and prepare it for deployment. Once the testing phase is finished, the code is deployed, and the work item is considered complete. Every work item must be processed at all three stations before it is complete.

4.2.2 Work Item Types

The work of the software development team is clustered into three types; type 1 representing front-end software development, type 2 representing back-end development, and type 3 representing system design.

- Type 1: Front-end software development, including user interface and aspects such as layout, appearance, and interactivity
- Type 2: Back-end development, involving the application's logic, database management, and integration with other systems.

- Type 3: System design includes identifying software components, defining their relationships and interactions, and determining the technologies and frameworks to be used

In the simulation model, new work items with an assigned type are generated randomly according to a distribution set derived from the original data. Each work item type has to be processed at all three stations by a resource that is capable of handling the specific work item type. Figure 5 shows the Kanban board with three work item types in three swim lanes. The work items wait in "To Do" column until an analyst has capacity to pull into analysis. Only then, the items are in progress.

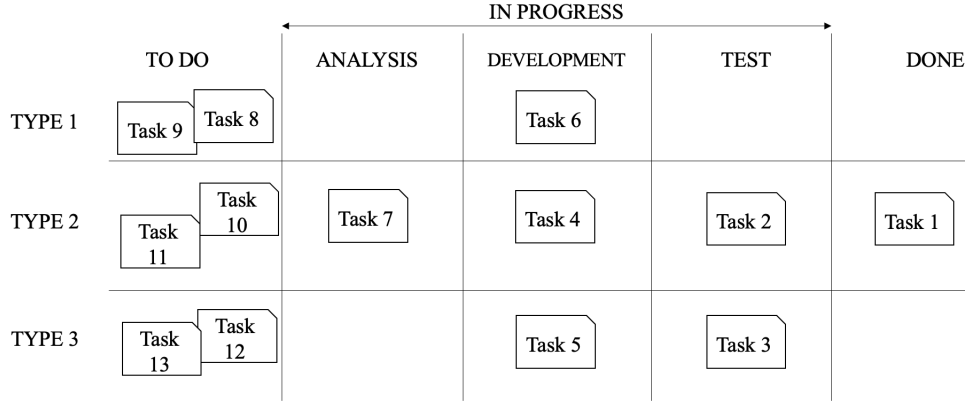


Figure 5: Kanban Board with 3 Work Item Types

4.2.3 Capabilities

Handling different work item types necessitates a diverse range of skill sets. Team members may possess a single capability to process one type of work or multiple capabilities to handle various work types. The incorporation of variations in capabilities introduces an additional level of complexity to the model. At each of the analysis, development, and testing stations, all three types of work must be carried out. To execute the work, the resource at the station must possess the requisite capabilities.

Each resource may have one of the seven capabilities, as illustrated in Table 2: Work Item Type - Capability Matrix.

Table 2: Work Item Type - Capability Matrix

Type - Capability	C1	C2	C3	C4	C5	C6	C7
Type 1	1	0	0	1	1	0	1
Type 2	0	1	0	1	0	1	1
Type 3	0	0	1	0	1	1	1

Resource capabilities in Table 2: Work Item Type - Capability Matrix include single capability resources (Capability types C1, C2, C3), double capability resources (Capability types C4, C5, and C6), and triple capability resource (Capability type C7). Modelling multi-capability resources (double and triple capabilities) requires a resource to handle more than one work item type at the same station. This arrangement results in a resource dividing its time between different work item types.

In the simulation, a day-based resource capacity structure is used to better represent the real-world scenario where each resource can work a full day or only part of the day, corresponding to 100% or 50% of capacity, respectively. This structure enables a more flexible allocation of resources based on their capabilities, allowing for a more accurate representation of the dynamics within the team. For example, a resource with capability type C4 can handle both type 1 and type 2 work and may divide its time between these two work types with a 50% split. This approach helps optimize resource utilization, ensuring that the team can adapt to varying workloads and types without sacrificing efficiency.

4.2.4 WIP Limits

Limiting Work-In-Progress (WIP) is a key principle of Kanban. Implementing this principle in the simulation requires introducing a "hold" functionality, which prevents work items from being processed unless the number of work items in progress is below the WIP limits. WIP limits help balance the workload and maintain a smooth flow of work items through the system, minimizing bottlenecks and reducing lead times.

In the simulation, a WIP limit parameter is defined that can be set at each simulation run. Before a resource pulls a work item, the simulation counts the total work items in progress and controls the WIP limit parameter. If there are less items than WIP limit in progress, the work item can be pulled. WIP limits can also be adjusted for each station to analyze their impact on the overall system performance. By experimenting with different WIP limits, the study aims to identify optimal configurations that maximize throughput while minimizing lead times and work item ageing. This exploration allows for a deeper understanding of the relationship between WIP limits and overall system performance, providing valuable insights for organizations seeking to improve their software development processes using the Kanban approach.

4.2.5 Blocking Scenario

In the complex landscape of software development, especially within large organizations, work items may encounter blockages due to factors outside the team's control. These blockages could arise from waiting for information, integration, or approval from another individual or department. When this occurs, teams following the Kanban approach typically place the work item into a blocked status and proceed with the next available work item. Once the blockage is resolved, the team resumes work on the blocked item as soon as possible, without sending it to the back of the queue. Figure 6 depicts a block happening to an item while being processed. When the block is solved, the item goes back to the queue, this time with higher priority so that it

will be complete before other items.

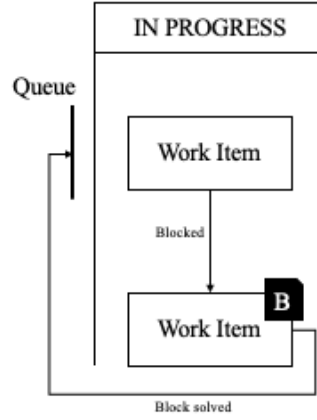


Figure 6: Blocking Scenario

In this simulation, a work item may be blocked at any of the analysis, development, and testing stations with a certain probability. The collected data indicates that block probabilities at each of these stations are independent of each other, meaning that a blockage at one station does not correlate to a blockage at any other station. The simulation software accounts for blocking scenarios by incorporating block probability, processing time before the block, block duration, and processing time after the block as parameters derived from the original data.

By simulating these blocking scenarios, the study aims to provide a more realistic representation of the challenges faced by software development teams in real-world situations. By analyzing the impact of these blockages on overall system performance, the study seeks to identify strategies for minimizing the negative effects of blocking scenarios, ultimately improving throughput and reducing lead times.

In addition to the features of the simulations described above, the simulation uses a set of base parameters regarding processing times. The base parameter set is explained in the following section.

4.2.6 Base Parameters

The base parameters of the simulation model are derived from the data collected from a large telecommunication organization in Turkey, which provides a realistic basis for the study. These parameters include processing and block times for each station, block percentages, and coefficients for work item types. The collected data has been thoroughly analyzed using the Input Analyzer feature of the Arena software [89], resulting in the following parameters.

Table 3: Processing and Block Times as Parameters

Station	Expression
Analysis no Block	LOGN(26.3, 67.6)
Analysis to Block	LOGN(32, 70.4)
Analysis after Block	53*BETA(0.438, 0.438)
Analysis Block	2 + EXPO(36.3)
Development no Block	WEIB(34.1, 0.988)
Development to Block	WEIB(31.2, 1.06)
Development after Block	LOGN(28.9, 59.1)
Development Block	2 + LOGN(24.3, 46.9)
Test no Block	EXPO(26.6)
Test to Block	0.999 + EXPO(16.1)
Test after Block	1 + EXPO(15.6)
Test Block	1 + WEIB(23.9, 0.703)

The block percentages represent the likelihood of work items being blocked at each station. Table 4 provides these percentages for the analysis, development, and test stations, based on the collected data.

Table 4: Block Percentages

Station	Block Percent
Analysis	18.45
Development	35.42
Test	19.59

Furthermore, the correlation between blocks at each station has been investigated. No significant relationship was found, with correlation scores of 0.049, 0.06, and -0.013 for the analysis-development, analysis-test, and development-test pairs, respectively. This implies that the occurrence of a block at any given station does not influence the likelihood of a block at another station.

The next set of parameters is the coefficient of processing times of work item types. These coefficients reflect the differences in processing times for the three work item groups: front-end development, back-end development, and system design. The front-end development work item type serves as the baseline with a coefficient of 1. Back-end development, with its slightly higher complexity, is assigned a coefficient of 1.25 for all processing times. The final work item type, system design, has a coefficient of 0.5. Although system design tasks can be highly complex, they are assumed to be broken down into smaller components in this study, mainly representing investigations or minor improvements. Consequently, system design work items require significantly less processing time compared to the other two types. It is important to emphasize that these coefficients are based on practical experience rather than being derived directly from the original data.

The final set of parameters relates to the output of the work item types. While each work item produces value, accurately estimating the value generated by a single work item is challenging since value realization often occurs after the completion of

multiple work items. In this study, work items are assigned to each type of work based on informed estimations, and the numerical analysis section presents multiple estimations using different values.

Overall, the initial simulation model is comprehensive and contributes to existing literature and research. It provides a realistic representation of the Kanban method in the context of software development. In the following section, the extended simulation model, and the additional complexity of the model is described.

4.3 Extended Model

The second level of the solution incorporates an enhancement to the simulation model and integrates it with machine learning algorithms and multi-objective optimization. In the initial model, resources are able to process one type of work at each station, and multi-capability resources split their time into different work types. Building on this initial model, in this extension, the resources are able to process multiple types of work according to their capabilities without splitting their time. All work items wait in the same queue until being pulled by a resource. The resources may pull different work items one by one in their working time according to their capabilities. The direct consequence of this model is prioritization of the work types.

4.3.1 Multi-Capability Resources

The explanation of work item type- capability relationship in Table 2 still holds. At each station, three types of work must be carried out by the analysts, developers, and testers. The new feature of multi-capability resources introduce a resource being able to process more than one type of work in working time without splitting the time. This feature is realized through a work item type - resource capability matching code snippet added to the processing block in the simulation. The effect of multi-capability resources on the computational scenarios is immense. For each station, instead of three types of resources processing the work, the possible type of resource

increases to seven representing each capability. The combination of each additional resource adds a multitude to the parameter universe of the simulation. This is the first driver of ML methods being investigated in the extended solution model.

4.3.2 Work Item Type Priorities

Once each resource is able to process multiple work item types, a new field comes out for deeper insights: priorities. The initial model only prioritizes work returning block, and the rest of the decisions are made through the first-in-first-out rule. In this extended simulation model, the resource may choose according to the priorities of each type of work. The priorities reflect the relative importance of work types, and are assigned through a numerical scale between 1, 2, 3. The existing high prioritization of work items returning from block is kept, and those items have the privilege to be processed before work items that have not been started.

Priorities have a direct impact on the complexity of the simulation strategy by multiplying the possible parameter combinations by the combinations of priorities for each work item type. The pursued path includes assigning a priority to each work item type independently, accumulating to three possibilities (1, 2, and 3) for each work item type, and 27 possible combinations in total simulation count. Given the potential simulation universe being already computationally challenging, multiplying by 27 leads to in-computable numbers.

Hence complexity, the simulation model allows the exploration of various mitigation strategies in response to blocking scenarios. These strategies may include adjusting WIP limits, reallocating resources, or also modifying work item priorities to ensure that the team can continue working on other tasks while waiting for the blockage to be resolved. The ability to simulate and analyze the impact of these strategies provides valuable insights for organizations seeking to optimize their software development processes in the face of unexpected challenges.

In conclusion, the simulation section lays the foundation for the study to provide a realistic estimation of the problem described in the previous section. The entire simulation code has been developed for this study using open-source Python libraries ensuring a high level of customization and adaptability. The blocking scenario incorporated into the simulations is a unique feature of the problem. The parameters, such as processing times and blocking percentages, are derived from a large telecommunication organization's data, enabling the simulation to effectively capture the nuances of the software delivery process. The simulations serve as a valuable contribution by capturing the software-specific dynamics of the Kanban system.

CHAPTER V

INITIAL SOLUTION MODEL

The initial solution model [2] consists of three integrated solution approaches; greedy Heuristic, two-step, and a decision tree clearing function (See Figure 7). The goal of these methods is to optimize resource allocation and WIP limits, thereby improving the team's efficiency, productivity, and ability to deliver high-quality work. The simulation model serves as a cornerstone of the solution approaches by estimating the results of different resource allocations and WIP limits. The initial simulation model features three types of work, three stations, the blocking scenario, and resources that are able to handle all types of work at all stations.

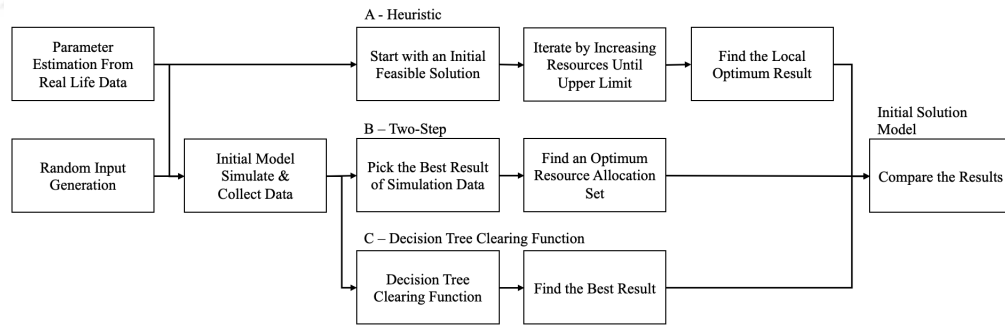


Figure 7: Initial Solution Model

The first part of the solution (A in Figure 7) combines simulations with a greedy heuristic algorithm. This approach incrementally develops an initial feasible solution until a capacity limit is reached. Offering a fast and straightforward approach, the greedy heuristic delivers competitive results. Upon completing the heuristic, a multi-capability resource allocation optimization is performed to determine a solution that satisfies the resource input of the heuristic result.

The two-step method (B in Figure 7) commences with an exhaustive simulation set of all possible resource allocation scenarios and finds the best spot. In the next step, the same multi-capability resource optimization is performed to determine a solution that satisfies the resource input of the best simulation result.

The final method (C in Figure 7) employs a decision-tree clearing function which uses a decision tree regression in conjunction with resource allocation, distinguishing itself by its ability to incorporate resource allocation challenges directly into the simulation results. This method provides a single integrated step whereas the first method is an incremental approach, and the second method consists of two steps.

The following sections explain the multi-capability resource optimization model which finds a resource allocation satisfying the simulation output, the greedy heuristic model, the two-step model, and the decision tree clearing function. The chapter ends with a numerical analysis of the solution approaches and compares the effectiveness of the proposed methods, highlighting their strengths and weaknesses.

5.1 Multi-Capability Resource Optimization Model

The complementary element of the initial solution model is the multi-capability resource optimization model. In the software development process, each work item must be processed by resources across all analysis, development, and testing stages to be completed. These resources include analysts, developers, and testers who handle work items of three distinct types: front-end (1), back-end (2), and system design (3). The model simulates each type of resources at varying levels. The resource levels start from 0.5 representing an employee working 50% of their time. For instance, a simulation may execute a run with 0.5 analysts capable of handling work item type 1, 0.5 analysts for type 2, and 0.5 analysts for type 3, totalling 1.5 analysts. The amounts of resources are incremented in the series of simulations up to the maximum limits determined. However, real-world software development teams are composed

of full-time equivalent team members, necessitating a more pragmatic approach to resource allocation.

To tackle this challenge, the multi-capability resource optimization model, which aims to devise a resource allocation solution satisfying the simulation results is developed. This optimization model strives to balance the theoretical and practical aspects of resource management to enhance the efficiency and effectiveness of software development teams. By adjusting resource allocations to accommodate multi-capability team members, this model ultimately contributes to improved project outcomes and overall team performance.

The mathematical formulation of the multi-capability resource optimization model is provided below:

Sets:

i = work item types, $i \in I = \{1, 2, 3\}$

j = stations, $j \in J = \{1 : \textit{Analysis}, 2 : \textit{Development}, 3 : \textit{Test}\}$

k = capabilities, $k \in K = \{1, 2, 3, 4, 5, 6, 7\}$

Parameters:

$t_{ik} = 1$ if capability k is able to handle work item type i , 0 otherwise

c_{jk} = cost of each resource at station j with capability k

Decision Variables:

s_{ij} = resource capacity input used on work item type i at station j

x_{jk} = # of resource of capability k at station j

y_{ijk} = amount of resource capacity of capability k on work item type i at station j

Constraints:

$$y_{ijk} \leq t_{ik}M \quad \forall i, j, k \quad (1)$$

$$s_{ij} \leq \sum_i y_{ijk} \quad \forall i, j \quad (2)$$

$$\sum_i y_{ijk} = x_{jk} \quad \forall j, k \quad (3)$$

The first constraint guarantees that capacities of resources will only be used based on defined capabilities. This constraint ensures that the resource capacity input doesn't exceed the total amount of resource capacity for each work item type, whereas the third constraint turns the capacities into full-time resources at specific capabilities at each station.

Objective Function:

$$Min \sum_{jk} x_{jk} c_{jk} \quad (4)$$

The objective is to minimize the total cost, which is the sum of the product of the number of resources and the cost of each resource. This model enables the optimal allocation of resources across work item types and stations, considering the capabilities of resources, while minimizing the total cost.

5.1.1 Parameter Values

The first parameter t_{ik} represents the capability of resource and work item relationship. t_{ik} is 1 if the resource of the according capability can process the type of work item, and 0 otherwise. Table 5 shows the seven capabilities of resources and which work item types they can process.

Table 5: t_{ik} ; capability of resource to handle work item types

	Type 1	Type 2	Type 3
Capability 1	1	0	0
Capability 2	0	1	0
Capability 3	0	0	1
Capability 4	1	1	0
Capability 5	1	0	1
Capability 6	0	1	1
Capability 7	1	1	1

The next set of parameters are the costs of the resources in x units. The costs represent all relative costs necessary to complete work item types and deliver value. Table 6 presents the costs associated with each resource capability for analysts, developers, and testers. It is important to note that the costs for developers are higher compared to analysts and testers, reflecting the relative scarcity of developers in the IT industry.

Table 6: c_{jk} ; cost of each resource capability

	Analyst	Developer	Tester
Capability 1	1000	2000	1000
Capability 2	1000	2000	1000
Capability 3	1000	2000	1000
Capability 4	1500	2500	1500
Capability 5	1500	2500	1500
Capability 6	1500	2500	1500
Capability 7	2500	3500	2500

As depicted in table 6, the costs for analysts and testers are the same for capabilities 1 through 3. Starting from capability 4, the costs for analysts and testers increase. For developers, the costs are consistently higher. For all resources, the cost of capability 7 is the maximum, since it represents a highly skilled senior employee capable of handling all types of work. This cost structure reflects the premium placed on developer resources and the need to optimize their allocation to maximize efficiency and cost-effectiveness in the software development process.

The last set of parameters enables the balance between the costs of resources and the generated value through the delivery of work items. The delivery of each work item corresponds to an output value according to its type. The output parameter is explained in Table 7. For the base simulation parameter set, the first work item type is 50 x units, whereas the second work item type is 100 and the third type is 60 x units. The output is the generated value of the team.

Table 7: Output Values

	Type 1	Type 2	Type 3
Value	50	100	60

The multi-capability resource optimization model bridges the gap between simulation-based estimations and real-world resource allocation scenarios for the Kanban system of the software development team. By considering the different work item types, stations, and capabilities of the resources, the model provides a practical and efficient approach to managing the allocation of full-time equivalent team members, while considering varying costs associated with each resource capability.

5.2 *Greedy Heuristic Model*

In the greedy heuristic model (simply the heuristic model), the simulations start from a minimum feasible resource allocation, which is 0.5 at each station for each work type. This accumulates to a total of 1.5 person analysis resources, 1.5 person development resources and 1.5 person test resources. Upon start, every resource is increased one by one, and each increase is simulated at four different WIP limits to find the best increment generating the highest output. The resource and WIP limit combination of the highest output is chosen as the best next move and the iteration ends with improving to the next base point where the next iteration begins. In the heuristic method, a maximum resource amount of 1.5 at each station is taken as the upper limit. The algorithm stops if any resource goes beyond the limit and takes the maximum value as the final value as shown in Figure 8.

After the iterations hit the upper limit and the simulation phase ends, the final parameter configuration is taken into the multi-capability resource optimization model to find the best allocation of resources. The result of the resource optimization

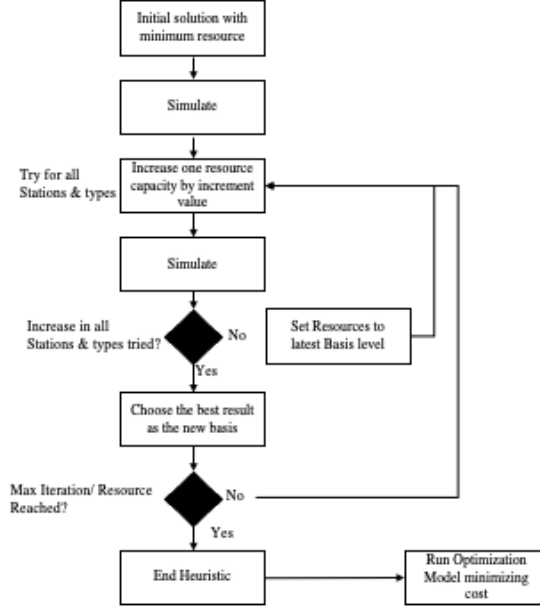


Figure 8: Diagram of the Heuristic Model [2]

model shows the cost of the resource setting. The generated output minus the cost of resources concludes the heuristic method.

5.3 Two-Step Model

The two-step model presents a more comprehensive methodology to address the problem by dividing it into two distinct stages. In the first stage, the model employs exhaustive simulations originating from a starting point and explores all possible combinations up to the maximum resource amount, factoring in various WIP limits for each resource configuration. Resource levels begin at 0.5 and increment by 0.5 until reaching 2, with four distinct WIP limits simulated at each level. The total number of simulations amounts to $4^9 \times 4 = 1,048,576$. Using the computer equipment with the Intel Core i5 processor, one simulation run took approximately 0.5 seconds, leading to the whole simulation cycle of 1,048,576 runs around 145 hours, or 6 days.

Upon completion of the first stage, the highest value among all simulations is identified. Subsequently, the resource configuration corresponding to this optimal

value is input into the multi-capability resource optimization model to ascertain the most efficient resource allocation that satisfies the best simulation's resource setting. Subtracting the cost of resources from the output provides the result of the model. Figure 9 shows the implementation steps of the two-step model.

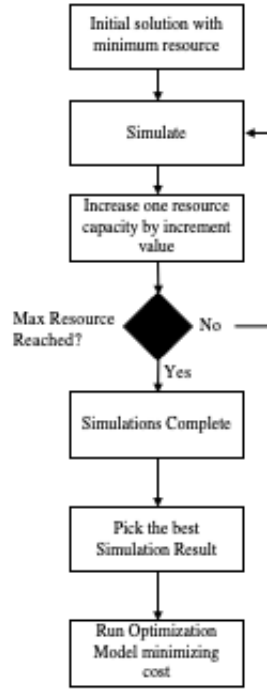


Figure 9: Diagram of Two-Step Model [2]

The main advantage of the two-step model is its comprehensive overview of the simulations which can directly identify the highest value of the simulations. However, this method is computationally more demanding, since it needs over one million simulations. Moreover, due to the fact that the simulations are decoupled from the resource optimization model, an optimum result is not guaranteed, which means the highest value of the simulations may come up with a higher cost, ending at a point lower than the actual optimum.

5.4 Decision Tree Clearing Function

The decision tree clearing function offers a more sophisticated approach, integrating simulations and the multi-capability resource optimization model through a clearing function. Like the two-step model, this method employs exhaustive simulations; however, instead of partitioning the simulation data from the optimization model, it utilizes all available data to train a decision tree regression. In this approach, each node within the tree signifies a decision point, such as allocating a specific resource to a station or establishing a particular WIP limit. The branches of the tree correspond to various potential decisions, while the leaves represent the ultimate outcomes. Consequently, the simulation results that exhibit a lower value than the optimal solution while remaining cost-effective are also considered. Such simulation results are largely overlooked in previous models. Figure 10 illustrates the implementation steps of the decision tree clearing function model.

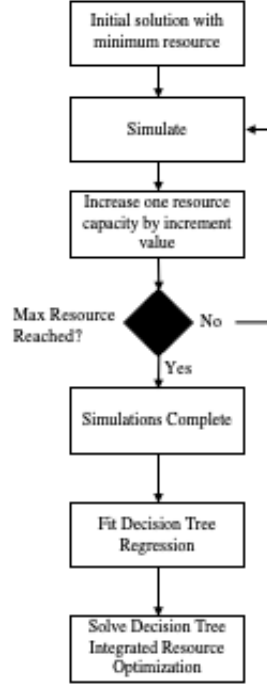


Figure 10: Diagram of Decision Tree Clearing Function Model [2]

5.4.1 Decision Tree Regression

The initial stage of the decision tree clearing function involves fitting a decision tree regression to the comprehensive simulation results. In this regression, the parameters (x) encompass the resources and WIP, while the objective (y) corresponds to the output. Figure 11 provides a preliminary overview of the decision tree, which shows higher values in darker and lower values in lighter colors. The decisions of the decision tree are with higher resources on the right and lower resources on the left side. Thus, the main take from figure 11 is that the branches on the right side have higher values than the ones on the left.

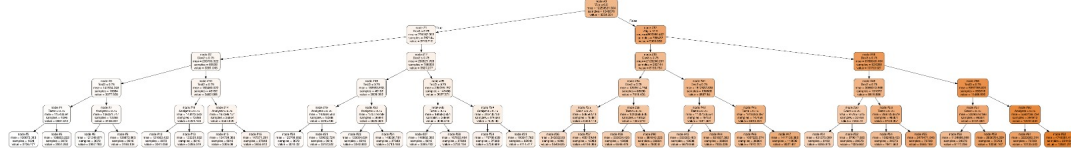


Figure 11: Decision Tree Regression Fit on the Simulation Results Max Depth: 5 [2]

5.4.2 Decision Tree Integrated Overall Mathematical Model

In this subsection, a novel model that integrates the decision tree clearing function with the multi-capability resource optimization model is introduced. This integrated approach aims to provide a more efficient and accurate solution for resource allocation in software development projects. The model considers various sets, parameters, and decision variables that pertain to work item types, stations, capabilities, features, and leaf nodes. The overall mathematical model of the decision tree clearing function is as follows:

Sets:

i = work item types, $i \in I = \{1, 2, 3\}$

j = stations, $j \in J = \{1 : \textit{Analysis}, 2 : \textit{Development}, 3 : \textit{Test}\}$

k = capabilities, $k \in K = \{1, 2, 3, 4, 5, 6, 7\}$

f = features, $f \in F = \{1, 2, 3 : \# \textit{ of analysts}, 4, 5, 6 : \# \textit{ of developers}, 7, 8, 9 : \# \textit{ of testers}\}$

m = leaf node, $m \in M$

Parameters:

$t_{ik} = 1$ if capability k is able to handle work item type i , 0 otherwise

c_{jk} = cost of each resource at station j with capability k

$mean_m$ = mean value of node m showing output

u_{fm} = upper bound of the decision variables representing feature f at node m

l_{fm} = lower bound of the decision variables representing feature f at node m

$output$ = output value

Decision Variables:

x_{jk} = # of resource of capability k at station j

y_{ijk} = amount of resource capacity from capability k on work item type i at station j

$z_m = 1$, if node m is active, 0 otherwise

r_f = value of the decision variable representing feature f

Constraints:

$$y_{ijk} \leq t_{ik}M \quad \forall i, j, k \quad (5)$$

$$\sum_i y_{ijk} = x_{jk} \quad \forall j, k \quad (6)$$

$$\sum_m z_m = 1 \quad (7)$$

$$output = \sum_m mean_m z_m \quad (8)$$

$$r_f = \sum_{ijk} y_{ijk} \quad \forall f \quad (9)$$

$$r_f \leq \sum_m u_{fm} z_m \quad \forall f \quad (10)$$

$$r_f \geq \sum_m l_{fm} z_m \quad \forall f \quad (11)$$

$$z_m \in \{0, 1\} \quad (12)$$

This model integrates the decision tree regression with the multi-capability resource optimization model. Thus, the first two constraints come from the multi-capability resource optimization model ensuring capacity usage based on capabilities (5) and capacity usage conversion to full-time equivalent resources (6). The following constraints (7) force only one decision node to be active and (8) describe the output as the mean value of the active node. The constraint (9) is linking the capacity usage y_{ijk} to the respective value of the decision tree r_f and the following two constraints

are upper (10) and lower (11) bounds. The last constraint declares that the decision variable for the nodes can only be 0 or 1.

Objective Function:

$$Max\ output - \sum_{jk} x_{jk} c_{jk} \quad (13)$$

The objective function aims to maximize profit, which is the difference between the output and the cost of resources. This reflects a balance between allocating more resources for maximal output and utilizing fewer resources for minimal cost. Through this integrated model, it is possible to optimize resource allocation decisions while considering resource utilization and excess resources.

5.5 Numerical Analysis

In this section, a numerical analysis is presented to compare the performance of the three solution methods proposed in this chapter: the heuristic, the two-step, and the decision tree clearing model. The analysis is based on a set of realistic scenarios and constraints, reflecting the challenges faced by the software development team described in the problem description.

The numerical analysis starts with the heuristic model, explaining the number of total resources, results, and improvements.

5.5.1 Heuristic Results

The first table below shows the incremental steps of the heuristic model including the base total number of resources, output value at the base (in x units), best improvement found at the station and the WIP limit at the best improvement found.

Table 8: Steps of the Heuristic Model

Step	Total no of Resources at Base	Output Value at Base	Best Increase found at	WIP Limit at Best Increase
1	4.5	2320	Developer Type 3	15
2	5	3730	Developer Type 1	21
3	5.5	4020	Test Type 2	15
4	6	4210	Developer Type 2	9
5	6.5	4550	Developer Type 1	15
6	7	4570	Analyst Type 3	15
8	7.5	5070	Test Type 3	15
9	8	5480	Test Type 1	21
9	8.5	5600	Analyst Type 1	21
10	9	5980	Developer Type 3	21
11	9.5	6680	Test Type 3	21
12	10	7390	N/A	N/A

The heuristic model runs for 11 steps and finishes at the 12th (See Table 8). Each step consists of trying out an increment of 0.5 at each of the 9 resource types and 4 different WIP limits(3, 9, 15, 21) at each of these resource increments. Thus, each step consists of 36 simulations and in total 396 simulations. The heuristic hits the maximum resource limit of 10 finding the highest output value of 7390 units. At this moment, the resource setting is shown in Table 9.

Table 9: Resource Setting of the Final Result of the Heuristic Model

	A1	A2	A3	D1	D2	D3	T1	T2	T3
Resource	1	0.5	1	1.5	1	1.5	1	1	1.5

This resource setting is the input for the multi-capability resource optimization model which finds a final solution with more than 10 resources, which is the maximum resource cap. Thus, the resource setting of the Heuristic model is infeasible. For this reason, the row before the last step is taken into account (Row 11 in Table 8), and the resource optimization model finds a resource allocation of 10 resources explained in Table 10.

The feasible resource allocation of the heuristic model is 10 resources, which is different than the initial model. For this reason, the new resource distribution is simulated again providing an output value of 12,320 at the cost of resources of 13,500 x units which accumulate for -1180 as the final value of the heuristic model.

Table 11: Resource Setting of the Best Result of the Simulations

	A1	A2	A3	D1	D2	D3	T1	T2	T3
Resource	1	1	1	1	1.5	1.5	0.5	1.5	1

Table 10: Heuristic Model - Resource Allocation

	Analysis	Development	Test
Capability 1	1	0	1
Capability 2	1	1	1
Capability 3	1	1	2
Capability 4	0	0	0
Capability 5	0	1	0
Capability 6	0	0	0
Capability 7	0	0	0
Total Resources	3	3	4
Total Cost	13,500		

5.5.2 Two-Step Model Results

The two-step model starts with the simulation phase consisting of incremental simulation of resources from 0.5 to 2 at each of the 9 resource types and four different WIP limits(3, 9, 15, 21), which accumulates to $4^9 \times 4 = 1,048,576$ simulations. The model finds the highest output with a maximum resource limit of 10. This value is 13,610 and found at 10 resources and a WIP value of 21. The resource setting of this maximum output is shown in Table 11.

Then, the method proceeds with the second step: multi-capability resource optimization. This step takes the resource setting of the highest output simulation

as an input and allocates the minimum cost resource capability distribution. The multi-capability resource allocation finds a result of 3 analysts (capabilities 1, 3, 6), 4 developers (capabilities 1, 2, and two times 3), and 4 testers (capabilities 2, two times 3, and 4) with a total cost of 16,000. Since the total output is 13,610, the final result is -2390 for the two-step model.

5.5.3 Decision Tree Clearing Function Results

The decision tree clearing function is a holistic approach combining simulations and resource optimization. It uses the data generated by the simulation set to train the decision tree regression and integrates it into the resource optimization model using upper and lower bounds.

In this study, the decision tree regressor function is used from the sklearn [90] package in Python. For the base results, a regression with a maximum depth of 5 is chosen. The R^2 value of the regression is 0.8383 which can be accepted as a strong result. Based on the decision tree with a maximum depth of 5, 320 upper and 320 lower bounds are found. The bounds are used as input parameters for the integrated optimization model.

The integrated optimization model finds an optimum solution of 1554.13 using the decision tree regression with a maximum depth of 5. A snapshot, showing the final levels of the decision tree that leads to the maximum output for the instance under consideration is shown in Fig. 12.

The decision tree clearing function finds the output value of 11,054.28 at the cost of 9500. At this node (see Fig. 12 right bottom node) there are 221,184 samples, which is an indicator of the need for further analysis of the decision tree clearing function. The multi-capability resource allocation finds a result of 2 analysts (capabilities 3, 4), 2 developers (capabilities 3, 4), and 2 testers (capabilities 3, 4) with a total cost of 9500 (See Table 12).

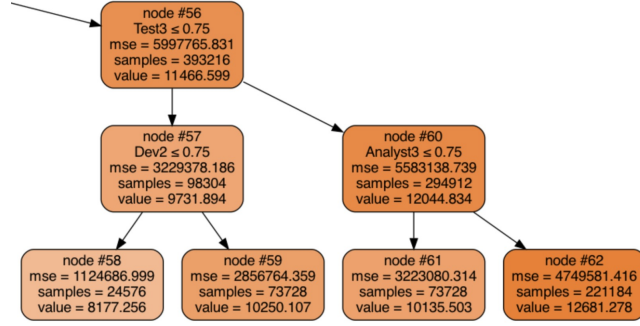


Figure 12: Decision Tree Regression Final 3 Nodes at Max Depth: 5 [2]

Table 12: Decision Tree Clearing Function Model Resource Allocation

	Analysis	Development	Test
Capability 1	0	0	0
Capability 2	0	0	0
Capability 3	1	1	1
Capability 4	1	1	1
Capability 5	0	0	0
Capability 6	0	0	0
Capability 7	0	0	0
Total Resources	2	2	2
Total Cost	9.500		

Since the output value is an aggregated value at the best node, it does not reflect an exact simulation result. Thus, the resource allocation is simulated again and the output value is found as 8500 at a WIP value of 21.

The final value is found as -1000 which is better than the benchmark models. The power of the decision tree clearing function model is shown in this case, which finds an optimum value at a total resource of 6 whereas the benchmark models find their

Table 13: Sensitivity Analysis Value and Cost Sets

Value and Cost Sets	Values Type 1, 2, 3	Resource Costs A1,A2,A3, D1, D2, D3, T1, T2, T3
Set 1	50, 100, 60	1000, 1500, 2500, 2000, 2500, 3500, 1000, 1500, 2500
Set 2	50, 100, 60	1000, 1300, 1500, 1500, 1750, 2000, 1000, 1300, 1500
Set 3	150, 200, 160	1000, 1500, 2500, 2000, 2500, 3500, 1000, 1500, 2500
Set 4	150, 200, 160	1000, 1300, 1500, 1500, 1750, 2000, 1000, 1300, 1500

best values at 10 resources. The reason for this is the decision tree clearing function method's ability to integrate the cost optimization instead of using two separate steps. The results of each model are compared and further insights are generated using a sensitivity analysis in the following section.

5.5.4 Sensitivity Analysis

The base results are merely objective and provide a biased view due to the chosen set of value and cost parameters. Also, in the study, the full simulation set of over one million data points is considered, whereas in practical cases, only a sample data set is accessible. For this reason, a sensitivity analysis is conducted. The sensitivity analysis covers three dimensions; the parameter set of values and costs, sample size, and decision tree depth. For the parameter set, two sets of values and two sets of costs are used. The values and cost sets (in x units) are shown in Table 13.

The second dimension of the sensitivity analysis is the sample size. In this study, 1,048,576 ($4^9 * 4$) simulations represent all combinations. Instead of trying four different levels of resource capacity (0.5, 1, 1.5, 2) at each work item type at each station, trying five would end up in 7,812,500 ($5^9 * 4$) simulations, whereas adding one more work item type would need 67,108,864 ($4^{12} * 4$) runs.

Table 14: Sensitivity Analysis

Value and Cost	Sample Size	Results		
		Heuristic	Two-Step	Decision Tree (5)
Set 1	1000	-1180	-3370	-1516
	10,000		-3460	-1000
	Full		-2390	-1000
Set 2	1000	-920	-940	234
	10,000		-930	650
	Full		10	650

The numbers can easily grow beyond computation limits. In such cases, since a full set of simulations is not available, the most favorable solution is to use sample data to conduct estimations. For this reason, data size is one of the dimensions of sensitivity analysis. In addition to the full set of simulations, two different sample sizes of 1000 and 10,000 are selected. For each of these sample sizes, five data samples are picked randomly. The results of the sensitivity analysis are presented in Table 14.

The results show that the decision tree clearing function outperforms the other methods in these value and cost combinations. It is seen that the heuristic method is very competitive at smaller sample sizes since it is not dependent on the sample size. Increased sample size improves the performance of the two-step method since the maximum value of the sample drives the method. Also, decision tree clearing function improves with an increase in sample size up to a level. Value and cost set one is totally cost dominated, in which the results are negative. On the other hand, value and cost set two is cost dominated for the heuristic method and the two-step method except for the full size. For this value and cost set, the decision tree method is capable of finding positive results. To test the methods under value dominated -profit- scenarios, a higher value set is tested with the same cost sets.

The results of the profit scenarios are shown in Table 15.

Table 15: Sensitivity Analysis - Profit Scenarios

Value and Cost	Sample Size	Results		
		Heuristic	Two-Step	Decision Tree (5)
Set 3	1000	16,490	14,442	12,054
	10,000		16,088	12,500
	Full		19,240	12,500
Set 4	1000	16,070	16,692	12,852
	10,000		18,278	14,150
	Full		21,440	14,150

The profit scenarios add up to the findings regarding the heuristic method. It performs competitively with the two-step model with smaller samples. The main highlight of the profitable scenarios is the decision tree clearing function. In these scenarios, the decision tree with depth of five performs worse. The last takeaway is the two-step method. Just like the cost dominated parameters, two-step method improves with larger sample sizes, and reaches to the highest results at full size. The profit scenarios show that the decision tree clearing function with depth of 5 is incapable of competing with the others. So, further analysis was conducted on these scenarios with increased depth of decision tree. In addition to 5, decision tree method with depths of 10 and 15 are conducted and shown together with previous results in Table 16.

Increased level of depth shows that the decision tree method is capable of finding better results with more depth. In both value and cost sets, decision tree results increase for the sample sizes of 10,000 and full size. For only value and cost set 3, decision tree at the depth of 10 and 15 cannot find a higher result than the depth of five, which indicates it is an exceptional case due to random samples.

Table 16: Profit Scenarios - Additional Decision Tree Depths

Value and Cost	Sample Size	Results		
		Decision Tree (5)	Decision Tree (10)	Decision Tree (15)
Set 3	1000	12,054	11,338	11,474
	10,000	12,500	13,116	15,624
	Full	12,500	13,190	15,380
Set 4	1000	12,852	14,178	13,960
	10,000	14,150	15,406	17,676
	Full	14,150	15,390	17,380

5.5.5 Findings

The value and cost parameters end up in negative results in the first two sets, and positive results in the latest two sets. The first observation by clustering the results in positive and negative is that the two-step model with a full data set finds the highest results in profit scenarios. The reason is, by its nature, the first step of the two-step model is finding the highest single simulation result of the model. This is a straightforward global maximum point. From this level, the model applies resource optimization to find the best fitting resource set. In case the model finds a resource allocation satisfying the highest value resource simulation, then the problem is solved. Only in the case that the resource optimization cannot find a satisfying result, another lower simulation result can be tried. The resource optimization model is flexible, and especially in positive cases, its contribution to the positive results is relatively small.

The other benchmark model, which is the heuristic, starts with an initial minimal data set and goes only until a local optimal point before conducting resource optimization. When compared to the global maximum value of the two-step model, it is only a matter of an unlikely coincidence that the heuristic model would find the global maximum. On the other hand, the power of the decision tree clearing function comes from integrating the costs of the resources and the generated value together

into the optimization model. This strength comes forward in cases where resource costs balance the values, and the final results are close to zero, or negative.

The second observation is based on the heuristic method. The heuristic method starts with an initial feasible solution which has minimum adequate resources and increases the resources based on the highest improvement. The method stops at a given limit of a maximum number of resources. Unlike the other two models, the heuristic model does not need any sample data. Hence, the method itself uses simulations to find the highest improvement at each node. In a case where the heuristic model ends at 11 steps as in the base scenario above, 396 simulations are conducted, where the minimum number of base data for the other two steps is 1000. Thus, the total number of simulations and necessary computation time is much smaller for the heuristic method. The results of the heuristic method are very competitive with limited data (sample size of 1000), and even for a relatively small amount of data (sample size of 10,000) the heuristic model can still be competitive. The heuristic model falls behind with an increased amount of data.

The decision tree clearing function provides the best results in all of the cost dominated sets. In these sets, the highest value is achieved through minimum cost. In most cases, the decision tree clearing function with the depth of five ends at a resource distribution of six resources in total. This is a conservative, low-cost solution.

Another significant pattern is that the decision tree clearing function results are improving in line with decision tree depth in profit scenarios. In both profit scenarios at sample sizes of 1000, and 10,000, the decision tree clearing function with the depth of 15 performs better than with the depth of 10. So, the decision tree gets better and better. Theoretically, if the decision tree goes to a depth that is enough to include only one result in a node, it should reach or pass all other methods, since it should find the exact best solution. But this case is not practically possible due to computation time at large sample cases and is not the intent of using the decision tree clearing

Table 17: Best Performing Method Based on Scenario Types

	Amount of Data		
	Small	Mid	Large
Profit Scenarios	Heuristic	Two-step/ Decision tree	Two-step
Cost Scenarios	Heuristic/ Decision Tree	Decision tree	Decision tree

function.

Table 17 summarizes the best performing methods based on scenario types and amount of data. Taking the summary and the results into consideration, as a conclusion of the numerical analysis, it is possible to come up with three key findings:

1. Two-step model performs best in profit scenarios. In both profit and cost dominated scenarios, the performance of the two-step model increases with the amount of data.
2. Heuristic model is competitive with limited data and computational time. When the information level increases, heuristic starts to be irrelevant.
3. Decision tree clearing function performs best in cost dominated scenarios. Also, when there is reasonably good and high amount of data, decision tree performs well. As seen in profit cases, the method's performance increases with higher level of depth.

CHAPTER VI

EXTENDED SOLUTION MODEL

The initial solution model includes a novel simulation module, two effective benchmark methods, and a decision tree clearing function that outperforms these benchmarks. However, its representation of multi-capability resources is limited to allocating resource time into intervals, whereas the true utility of multi-capability involves processing different types of work sequentially. The extended capability integrated simulation model addresses this limitation by enhancing the resource's ability to handle multiple types of work.

6.1 Capability-Integrated Simulation Model

The capability-integrated simulation model introduces resources capable of processing more than one type of work sequentially within the same time interval. For example, a resource with the capability to handle all three types of work (type 7) can process work item types one, two, and three one after another. This adds a layer of complexity in determining which type of work to process first. The default decision-making mechanism is a first-in-first-out order, enhanced by an additional priority for items returning from a block. This approach could significantly impact output of the team and requires deliberate decision-making to optimize results. Therefore, the model implements specific priorities for each type of work (1, 2, or 3), with resources prioritizing work items based on these priorities. This priority system retains the initial added priority for returning work items. Figure 13 illustrates how the initial and extended simulation models differ in processing work item types and setting priorities.

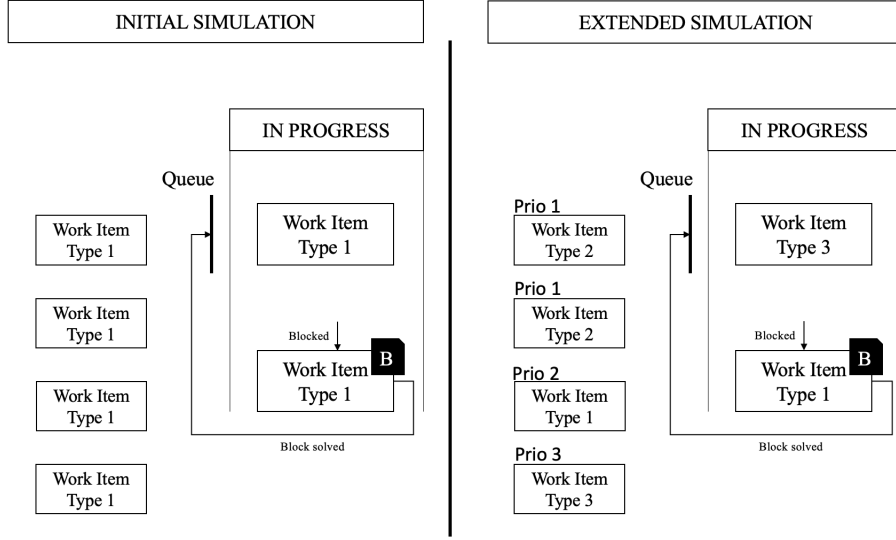


Figure 13: Initial vs Extended Simulation Model

The enhancement of the simulation model with the integration of the capabilities has two direct consequences on the overall solution approach:

1. The number of possible simulations
2. The resource optimization model

6.1.1 The Number of Possible Simulations

The initial simulation model employs three types of resource processing each work type at each station, and implements four levels of WIP limits totaling 1,048,576 ($4^9 \times 4$) possible configurations. The capability-integrated simulation model embeds different capability resources directly. For instance, capabilities 1, 4, 5, and 7 can all process work type 1 (refer to Table 2). Similarly, to process all three types of work, three types of resources were used in the initial simulation model, whereas in the capability-integrated model, a combination of resource capabilities is necessary. While choosing this resource combination, due to simplification, only one capability of resource is chosen that can process one type of work. As an example, only one of the capabilities

1, 4, 5, 7 can be part of a solution combination since they can all process type 1 work. This results in five distinct preferred combinations valid at each station, detailed in Table 19.

Table 18: Preferred Solutions

Preferred Solution	Resource Capabilities						
	C1	C2	C3	C4	C5	C6	C7
1	1	1	1	0	0	0	0
2	1	0	0	0	1	0	0
3	0	1	0	0	0	1	0
4	0	0	1	1	0	0	0
5	0	0	0	0	0	0	1

In this capability-integrated simulation model, the initial resource amounts are kept, starting from the same 0.5 resources with 0.5 increments up to 2 resources presenting four different resource levels. To calculate the total number of combinations, the first step is looking at a single station. Work is processed at each station by resources from one of the preferred solutions at one of the different amounts. If the preferred solution is 1, there are three resources each of whom can take one of 4 resource amounts, which equal to $64(4^3)$ combinations. Table 19 depicts the total combination amount of 116, valid for each station. Considering analysis, development, and test stations, total combinations equal to $(116^3 = 1,560,896)$.

Table 19: Preferred Solutions

Preferred Solution	Number of Different Resource Capabilities	Resource Combination	Total
1	3	(4^3)	64
2	2	(4^2)	16
3	2	(4^2)	16
4	2	(4^2)	16
5	1	(4^1)	4
Total			116

In addition to resource combinations, the critical features of WIP limit and priority are also tested in the capability-integrated simulation model. As before, there are four different WIP limits, and in addition, there is a priority for each type of work. The prioritization is a relative priority to the other types of work, with the goal of determining which work item to start processing. Thus, each work type may have one of the priorities 1, 2 or 3, but not the same. This leads to six different combinations of priorities.

Calculating the total number of combinations for all simulations, the number of resource combinations should be multiplied by the number of WIP limits and the priority combinations to reach a staggering amount of over 37 million combinations: $116^3 \times 4 \times 6 = 37,461,504$. Given that the number of simulations is 1,048,576 in the initial solution model, the integration of capabilities together with the priorities leads to an increase of 35.7 times. Change in resource capabilities and prioritization of work item types is conducted through an additional code snippet acting as a decision-making module for each station before processing. Although it is a relatively short code snippet, it introduces a priority-based decision to all stations for every decision

before processing; this change created a significant increase, resulting in a single simulation time of 0.9 seconds, up from 0.5 seconds. Running 37,461,504 simulations would require over 390 days using the same hardware, making it computationally impractical.

6.1.2 The Resource Optimization Model

The second consequence of enhancing the simulation model impacts the multi-capability resource optimization model. The results of the initial simulation model show the amount of time spent on each type of work, and this time allocation of resources is solved in the following optimization model. Integrating these capabilities directly into the simulation model already demonstrates the results of specific capabilities, leaving no room for further allocation to capabilities. As a result, the multi-capability resource optimization model becomes redundant.

The resource optimization model serves as a complementary component for the heuristic, two-step, and especially the decision tree clearing function. The integration of capabilities to the simulation model disrupts all these methods and creates the urge to extend the solution approach. Due to this seemingly small but significantly impactful change in the integration of capabilities, the extended solution model explores new paths to achieve optimal resource distribution.

6.2 Extended Solution Model

The capability-integrated simulation model enables far more insights than the initial model through each capability of resources being input features of the model as well as work type priorities. To overcome the challenge of impractical computational time, the extended solution model leverages this additional power of the simulation through ML models.

ML is the leading way for exploring large datasets without predefined data structures or formulas. The models are trained using a viable amount of data, and can generate strong predictions for the possible event horizon without simulations. Accordingly, the extended model starts with capturing data from the capability-integrated simulation model from a random set of parameters. Three ML algorithms are trained using the data; random forest, XGBoost, and NN. Then, each algorithm predicts the full set of input combinations, and the best prediction results are simulated again. The comparison of the simulation results shows the best-performing ML algorithm (Figure 14). The extended solution model reaches a compatible result in an effective way.

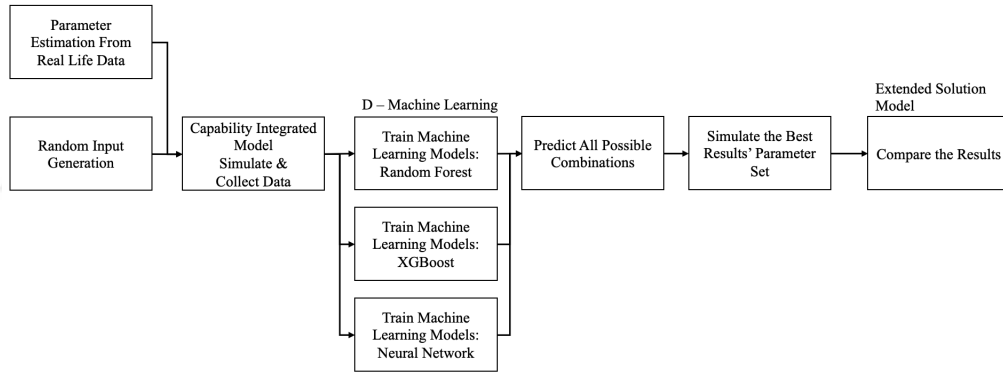


Figure 14: Extended Solution Model

6.3 *Machine Learning Models*

In the extended solution model, three prominent ML algorithms —random forest, XGBoost, and neural networks— are employed to analyze and predict outcomes from complex simulation model. Using multiple models enhances the strength of the research, and increases the ability to reach high prediction accuracy. This section provides a brief overview of each algorithm and delves into the construction and analysis of the extended solution.

The first ML model random forest is an ensemble learning technique that operates

by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. This method is highly effective for complex datasets because it reduces overfitting by averaging multiple trees. Each tree in a random forest works on a random subset of features to decide on a split, adding to the diversity of the decision-making process. The robustness of this model makes it a suitable choice for predicting outcomes based on diverse input features from the capability-integrated simulation model.

XGBoost, or Extreme Gradient Boosting, is an implementation of gradient-boosted decision trees designed for speed and performance. It is distinguished by its capacity to do both parallel processing and tree pruning. The latter refers to the reduction of branches that have little power to explain the variance in data. XGBoost also handles missing data internally, making it resilient and effective when dealing with incomplete data sets. It is particularly favoured for its high performance and fast execution, which are crucial in handling the expansive datasets generated from the simulations. By sequentially building trees, each new tree in XGBoost helps to correct errors made by the previous ones, enhancing the predictive accuracy as more trees are added.

Finally, NN is a class of machine learning algorithms modeled loosely after the human brain. They consist of layers of interconnected nodes or neurons, where each connection represents a weight that is adjusted during training. Neural networks are particularly versatile in handling non-linear data and can be adapted to various types of data input. This capability makes NNs extremely useful for the dataset obtained from the capability-integrated simulation model, as they can model complex patterns and interactions that may be missed by more traditional algorithms. The flexibility to design various architectures allows for extensive experimentation to achieve the best performance for the specific features of the dataset.

Together, these models provide a comprehensive approach to tackling the predictive challenges presented by the simulation data, with each model offering unique strengths that enhance the overall predictive capability of the extended solution model.

ML training and predictions are conducted in Python using the Sklearn [90] and Keras [91] libraries. Training ML algorithms start with selecting a feature set. The feature set consists of the parameters of the simulation which are analysts (a1, a2, .. a7), developers (d1, d2, .. d7), testers (t1, t2, .. t7), priorities (p1, p2, p3), and WIP limit with a total of 25 features. The target variable for prediction is the output representing the number of total completed work items. To obtain the training data, 5000 simulations are run with random variations of the selected features set which has 0.5 to 2 resources, four different WIP limits(9, 15, 21, 27), and three possible priorities (1, 2 or 3). The total amount of resources in the training data does not exceed 10, since the first goal is to find the maximum output with up to 10 resources. Once the simulations are complete, training data is collected.

The next step of training is splitting the 5000 data points into training and testing sets with a random allocation. Training set sizes are 20% for random forest and XGBoost, and 50% for the NN. The second step is feeding the model with the training set, and testing with the testing set. The testing phase leads to the performance evaluation of the ML models. The configurations chosen for each of the ML methods are as follows:

- Random forest model incorporates 100 trees where each tree is an individual decision tree.
- XGBoost also has 100 estimators (trees) and a 0.008 learning rate. The learning rate controls how much the contributions of each tree are scaled down before adding them to the final model.

- NN uses exponential linear unit (elu) activation, and 100 epochs. Activation functions are the functions of NN methods that address node to node transitions in the data, whereas epoch refers to one complete pass of the training set. Using more epochs improve fine tuning and the prediction capacity of the model.

After training the ML models using the respective libraries in Python, the ML models are evaluated based on the R^2 score and the mean squared error (MSE). The R^2 score indicates the proportion of variance in the dependent variables that is predictable from the independent variables while the MSE measures the average of the squares of the difference between the estimated and the actual.

Table 20 shows R^2 values of three ML methods trained using the data set generated by the simulations. Although all ML models are strong, with their R^2 values bigger than 0.75, XGBoost and NN are stronger than the random forest model, and very close to each other.

Table 20: Machine Learning Model Comparison

	R^2	MSE
Random Forest	0.75	320.5
XGBoost	0.836	213.8
Neural Network	0.837	214.1

6.3.1 Predictions up to 10 Resources

The power of ML lies in its capability to learn complex structures and provide effective predictions. In this case, ML algorithms are not provided with any formula or information regarding the flow and the dynamics of the team working with Kanban, but only with a feature set and 5000 data points generated by the simulations. In this step of the extended solution model, the ML algorithms predict output given all possible feature combinations. The feature combinations include the whole universe

of the training data with the total number of combinations being 37,461,504 whereas applying a resource limit of 10 reduces the total number of potential combinations to 24,759,216. Each ML algorithm is used to predict the output of over 24 million different input combinations, which take 1.5 minutes for random forest and XGBoost, and 6 mins for the NN using the same hardware. This computation accounts for approximately 275,000 predictions per second for random forest and XGBoost and 70,000 predictions per second for the NN. The highest output of each of the ML algorithms is taken as the best result, and the feature combination of this result is simulated to compare. The highest outputs, respective input levels and the simulation outputs are shown in Table 21.

Table 21: Machine Learning Predictions

	Best of ML Prediction	Input Features	Simulation Result of the Best Prediction
Random Forest	199.7	a1:1.5 a2:0.5 a3:1.5 d3:1.5 d4:1.5 t1:0.5 t2:1.5 t3:1.5	180
XGBoost	208.3	a3:1 a4:1.5 d1:2 d5:2 t3:2 t4:1	219
Neural Network	351.4	a3:2 a4:0.5 d1:2 d2:0.5 d3:2 t3:2 t4:1	212
All three ML predictions have the same priority and WIP values: p1:2 p2:3 p3:1 and WIP:27			

As Table 21 depicts, the best-performing ML model is XGBoost by finding a feature combination that leads to a simulated output of 219. The prediction of XGBoost is 208.3, which is 5% close to the simulation result. NN follows XGBoost, with a simulated output of 212, hence having a prediction of 351.4. The difference of 40% is more than expected for the NN which requires further investigation. Lastly, the random forest predicts 199.7 and the simulated output ends up as 180, which is within 10% range. random forest's prediction and simulation result are both the lowest of all three ML models, but prediction accuracy is better than the NN.

ML models prove their capabilities by finding accurate input sets out of the huge feature universe of 24,759,216 combinations.

6.3.2 Exploring Resources up to 15

A key strength of ML is its ability to explore further reach using limited training data. In this study, the ML algorithms are trained with only 5000 data points which consist of resources up to 10, however, the ML models are able to make effective predictions for combinations with more than 10 resources. For this purpose, the best-performing ML model -XGBoost- is chosen to make estimates for an extension of resources up to 15.

Table 22: Prediction and Simulation Results up to 15 Resources

Max Resource	Best of XGBoost Prediction	Simulation Result	% Change	Output per Resource
10	208.3	219	NA	21.9
11	217.7	221	0.9%	20.1
12	226.2	224	1.4%	18.7
13	226.4	232	3.6%	17.8
14	228.9	253	9.1%	18.1
15	229.9	268	5.9%	17.9

Table 22 shows the best of XGBoost predictions, the simulation result as the output of the feature combination of that prediction, the increase of the output to the previous output, and output per resource for each amount of resource from 10 to 15. The observations taken from this exploration are listed below:

- ML predictions and simulation results are constantly increasing which shows that ML methods provide consistently effective predictions for feature combinations beyond training data.
- The best output per resource is at the resource amount of 10. Even if the total output increases up to 268, the efficiency of the resources does not show the same increase.
- Although the predictions of XGBoost are still successful by leading to increased simulation results, they become increasingly undershooting with the higher amount of resources. The reason of the difference lies in the training data consisting of combinations of up to 10 resources.

In real-life cases, organizations may choose to increase their resources at any point. A great advantage of ML helps decision making without another round of training. Table 22 depicts the results of the method applied in this study for up to 15 resources. First, for each resource limit of 11, 12, 13, 14, and 15, all possible parameter combinations of resource capabilities, priorities, and WIP limit are generated. Then, all combinations are predicted using XGBoost as the best-performing ML algorithm. Finally, best parameter configuration of the ML algorithm is simulated, and the output is noted. Based on the results, an increase in resources beyond the base number of ten yields varying results at. Notably, the application of the XGBoost algorithm yields a downturn in output with eleven resources, registering a 15% reduction. However, predictions stabilize and align more closely with actual outputs as resources increase from twelve onwards. The data suggests diminishing returns on output with each added resource, indicating a non-linear relationship between resource input and productivity gains. This underlines the complexity of the Kanban model, and the importance of the methodology applied in this study in decision making. Simply, according to the increase of output gained at each resource level, the organization can conduct a profitability analysis and make the right decision.

6.4 *Insights*

In conclusion, the extended solution model presents an analysis and an optimized feature set for a more complex work structure simulated by the capability-integrated simulation model where the methods of the initial solution model become obsolete. For this purpose, the model integrates simulation data into three ML methods: random forest, XGBoost, and NN. Each ML model makes 24,759,216 predictions in a computationally effective way, and the best predictions are proved using the simulations. Furthermore, the best-performing ML model -XGBoost- is used to predict output for a feature set with resources up to 15. This analysis shows that the amount

of resources increase the output, but the most effective resource set is at 10 resources with the output per resource amount of 21.9. Overall, the extended solution model takes the optimization approach of this study to a next level, and provides data-driven decision making insights for organizations facing the challenge of resource allocation for teams working with Kanban.



CHAPTER VII

MULTI-OBJECTIVE OPTIMIZATION

The flow of work in Kanban has a complex structure with multiple decision points including work type classification, type and amount of resources, WIP limits, etc. In previous chapters, maximizing output with a minimum amount of resources has been explored using an approach based on simulations. Various techniques that combine the simulations with heuristics, resource optimization, clearing function, and ML methods were investigated. Hence, in Kanban systems, lead time is a crucial dimension, since the speed of value delivery is a key competitive advantage for organizations. In this chapter, the impact of lead time is investigated.

The integration of lead time into the framework aims to holistically improve the flow of work by not only enhancing productivity and cost-efficiency, but also by reducing the time taken from work initiation to completion. The introduction of lead time as an additional dimension of optimization presents a multi-objective optimization problem, which calls for a methodological approach to reconcile competing objectives. The MOO model is pivotal in identifying the most efficient resource allocations that balance productivity, cost, and time.

7.1 Multi-Objective Optimization Approach

The diagram (Figure 15) outlines the approach used to explore MOO. As in the extended solution section, MOO uses the estimated parameter set, and the capability-integrated simulation model. This simulation model produces a rich dataset that captures the outcomes of diverse configurations. The broad universe of possibilities are explored via integrating the XGBoost predictions to three MOEA models: SPEA-II, PAES, and NSGA-II. Finally, the non-dominated results are estimated, and

analyzed.

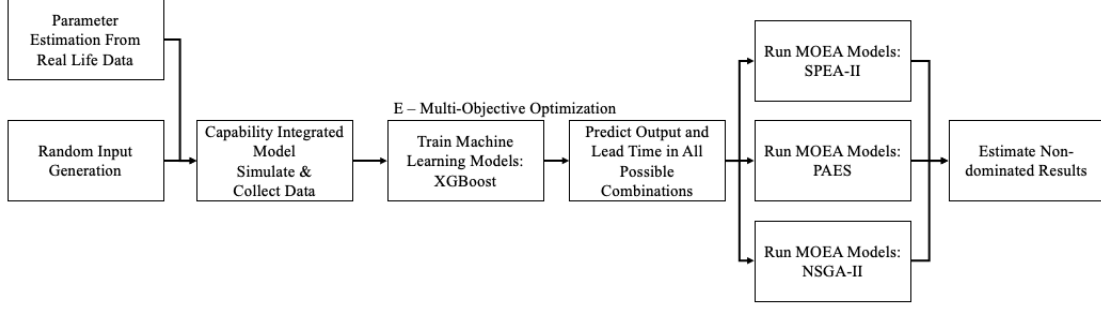


Figure 15: Multi-Objective Optimization Approach

7.1.1 Formulation of Multi-Objective Function

One of the solution paths of the MOO is the combination of objective functions into a single objective function. The mathematical formulation of a typical MOO objective function can be represented as follows:

Objective Function:

$$\text{Min} \sum_{i=1}^n a_i, \sum_{j=1}^m b_j, \sum_{k=1}^p c_k \quad (14)$$

Where a_i , b_j and c_k denote three different objectives, and combining them in a single objective function is an application of the scalarization method. The challenge of the method is allocating weights to the different objectives from different dimensions and bringing them to a combined function. For instance, the scope of this research includes a MOO problem regarding lead time, output and cost. Hence it may be possible to calculate the value of the output and assign a weight to compare with cost, it is extremely challenging to sum it up with lead time, and find a meaningful weight. For this reason, this research pursues evolutionary algorithms instead of scalarization methods.

7.2 Multi-Objective Evolutionary Algorithms

Exploring the vast space of 37,461,504 potential combinations, the use of a robust ML algorithm becomes indispensable. For this purpose, XGBoost is selected due to its superior performance over other algorithms like random forests and neural networks in previous assessments. In this comprehensive analysis, XGBoost is employed not only to predict the output but is also trained separately to estimate lead times. The results from these training sessions using a dataset of 5,000 randomly generated data points are displayed in Table 23. Notably, the model achieves a relatively lower R^2 score of 0.728 for lead time predictions compared to 0.836 for output predictions, highlighting the challenges in accurately forecasting lead times.

Table 23: XGBoost Performance at Output and Lead Time

XGBoost	R^2	MSE
Output	0.836	213.8
Lead Time	0.728	25,591,652

The predictions derived from the ML algorithm lead to a dataset comprising 37,461,504 data points, each characterized by a set of features, predicted output, and lead time. For the purpose of MOO, a Pareto analysis might be conducted for the three targets of lead time, output and total number of resources. Hence, like many MOO problems being NP-hard [92], such an analysis for the large data set in this problem is computationally impractical. For this reason, three different MOEAs are applied using the distributed evolutionary algorithms in python [93].

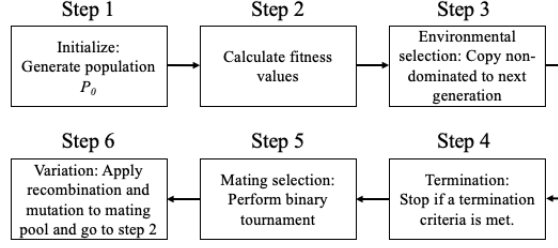


Figure 16: SPEA-II

7.2.1 Strength Pareto Evolutionary Algorithm-II

SPEA-II starts with generation of an initial population P_0 [75]. Next step is the calculation of a fitness function based on dominance on other members of the population. At the third step, an environmental selection is conducted that copies the non-dominated members of P_t to $P_t + 1$ and the fourth step checks if the number of generations, or another termination criteria is met. The fifth step performs a binary tournament for mating, and the last step applies recombination and mutation operations and sets $P_t + 1$ to complete the iteration and move to step two for the next iteration as shown in figure 16.

7.2.2 Pareto Archived Evolution Strategy

PAES [76] begins with a randomly selected initial solution, which undergoes mutation to generate a new candidate. This candidate is assessed for dominance against the current solution; if dominated, it is discarded and replaced by a new mutation. If not dominated, it undergoes further comparison against members of an archive before being added to it. This iterative process continues, adding candidates to the archive and generating new solutions until a termination criterion is met, as shown in Figure 17. One distinctive feature of PAES is its lack of a fixed population size, focusing instead on progressively building an archive of optimal solutions. The archive stores the non-dominated solutions, and it helps the selection process during the iteration

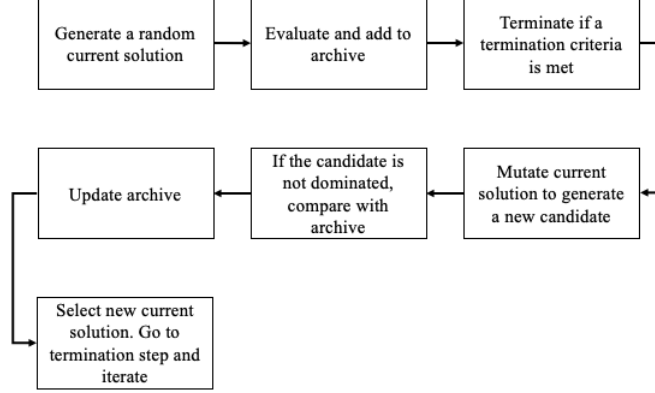


Figure 17: PAES

which is the main strength of it.

7.2.3 Non-dominated Sorting Genetic Algorithm-II

NSGA-II [77] initiates by selecting a group of members which are then sorted into dominating (P_t) and non-dominating groups (Q_t). Members are further classified into ranks based on their level of domination. The most dominant members are carried over to the next generation $P_t + 1$, while others are evaluated through a measure designed to maintain diversity within the population called crowd distancing. The subsequent generation is formed by retaining half of the members through these criteria and introducing the other half through mutation and crossover, ensuring a balanced approach to maintaining diversity and optimizing fitness, as illustrated in Figure 18.

7.2.4 Numerical Analysis

The numerical analysis of the MOO focuses on comparing the performance of SPEA-II, PAES, and NSGA-II under various parameters outlined in Table 24. This comparison creates a chance to assess how each algorithm yields different results in population size, number of generations, and mutation probability. SPEA-II and NSGA-II are tested with a consistent population size of 50, while PAES, uniquely, does not use

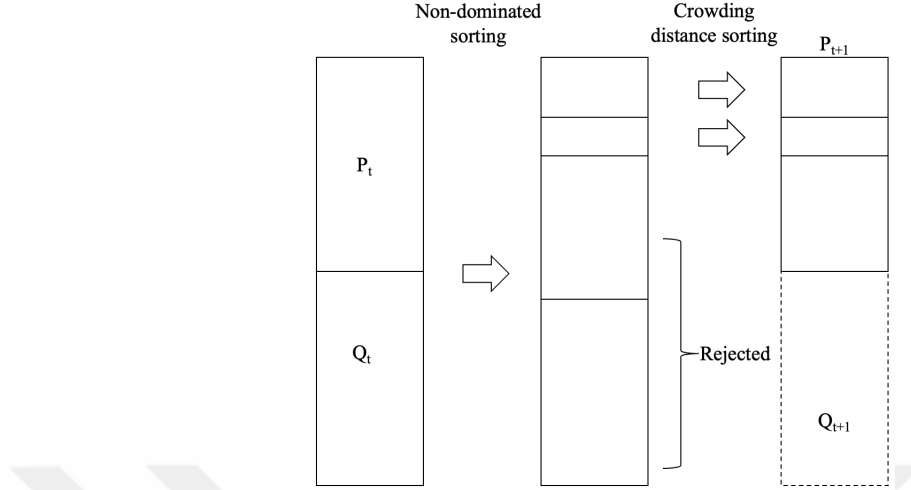


Figure 18: NSGA-II

a fixed population size, instead building solutions iteratively. All three algorithms undergo experiments over a range of generation settings—100, 500, 1000, and 3000—to examine their performance stability and scalability. Mutation probabilities are also varied at 0.2 and 0.4 to investigate their impact on the diversity and convergence of solutions. The outcomes of these tests are critical for understanding how different parameters influence the efficiency and effectiveness of each algorithm in navigating complex optimization landscapes.

Table 24: Parameters for SPEA-II, PAES and NSGA-II

	SPEA-II	PAES	NSGA-II
Population Size	50	NA	50
Number of Generations	100, 500, 1000, 3000		
Mutation Probability	0.2 and 0.4		

Upon completion of MOEAs, solutions obtained from each algorithm are aggregated, resulting in a total of 1012 results as displayed in Table 25. Subsequently, a

non-domination filtering process is implemented to refine these results by identifying which solutions are non-dominated. The non-domination algorithm operates as follows:

A solution is considered dominated if there is another candidate that matches or exceeds it in all objectives (lead time, output, total resources) and surpasses it in at least one. Specifically, a solution row is dominated if a candidate meets the following conditions:

- The candidate's lead time is less than or equal to that of the row,
- The candidate's output is greater than or equal to that of the row,
- The candidate's total resources are less than or equal to that of the row,

Importantly, the candidate must be strictly better in at least one of these metrics. Application of this algorithm to the collected data yields 52 non-dominated solutions. The distribution of these optimal solutions reveals that 46% are derived from PAES, while both SPEA-II and NSGA-II account for 27% each. This distribution indicates that PAES exhibits a superior performance in generating non-dominated solutions compared to the other algorithms.

Table 25: Aggregated Results of MOEA

	SPEA-II	PAES	NSGA-II	Total
All Solutions	400	212	400	1012
Non-Dominated Solutions	14	24	14	52
Percentage in Solutions	27%	46%	27%	100%

The non-dominated results are visualized in three-dimensional scatter diagrams from three different angles in figure 19. PAES results are represented in blue, while

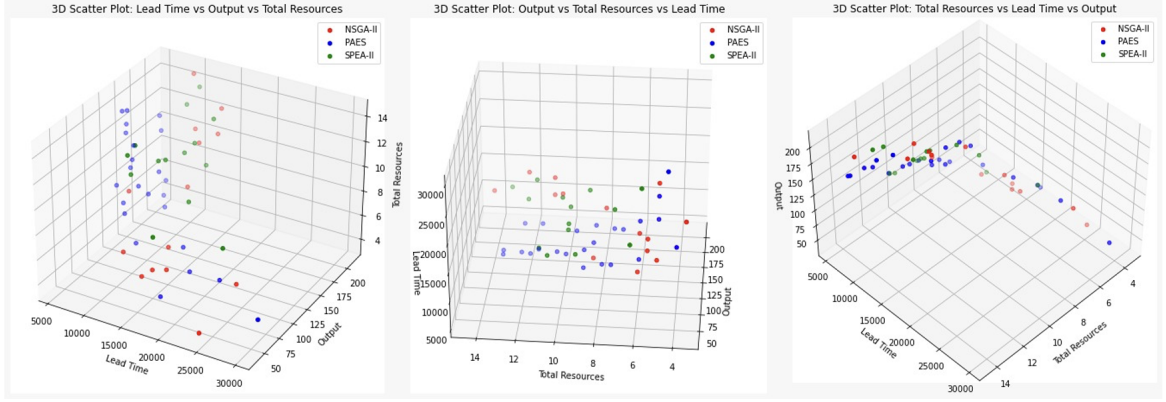


Figure 19: 3D Scatter Diagram of Aggregated MOEAs

SPEA-II in green and NSGA-II in red. This analysis underscores the robustness of PAES in producing solutions that remain resilient against domination in the multidimensional space of this optimization challenge, offering valuable insights for strategic decision-making in complex scenarios. However, further investigation is necessary to understand the performance of SPEA-II and NSGA-II.

The analysis extends to examining the impact of the mutation probability parameter on the effectiveness of different MOEAs. The results, as illustrated in Table 26, demonstrate that with a mutation probability of 0.2, SPEA-II accounts for 33% of the solutions, while NSGA-II contributes 21%. In contrast, increasing the mutation probability to 0.4 results in NSGA-II producing 30% of the solutions, surpassing SPEA-II. Despite these shifts, PAES consistently outperforms both algorithms at both mutation rates, generating 46% and 41% of the solutions respectively. This pattern suggests that PAES and SPEA-II are more effective with a lower mutation rate, whereas NSGA-II shows improved performance with a higher mutation probability.

Table 26: Impact of Mutation Probability on MOEAs

	SPEA-II	PAES	NSGA-II	Total
Mutation Probability: 0.2	13	18	8	39
Percentage in Solutions	33%	46%	21%	100%
Mutation Probability: 0.4	8	17	16	41
Percentage in Solutions	20%	41%	39%	100%

In finalizing the comparison among the MOEAs, the influence of the number of generations on algorithm performance is analyzed. As detailed in Table 27, at lower generation counts (100 and 500), both SPEA-II and NSGA-II exhibit superior performance, averaging 38% and 41% of solutions, respectively. It is noteworthy that PAES produces significantly fewer results at these generation levels when compared to the fixed population size of 50 used by the other methods. However, as the number of generations increases to 1000 and 3000, the domination of PAES also rises. At the highest generation count of 3000, the distribution of solutions among the three algorithms reaches its most balanced state, indicating an equilibrium where all methods are comparably effective.

Table 27: Impact of Number of Generations on MOEAs

	SPEA-II	PAES	NSGA-II	Total
Number of Generations: 100	14	6	12	32
Percentage in Solutions	44%	19%	38%	100%
Number of Generations: 500	12	10	17	39
Percentage in Solutions	31%	26%	44%	100%
Number of Generations: 1000	5	15	10	32
Percentage in Solutions	22%	47%	31%	100%
Number of Generations: 3000	9	13	12	34
Percentage in Solutions	26%	38%	35%	100%

The comparative analysis of SPEA-II, PAES, and NSGA-II across varying mutation probabilities and generation counts reveals distinct performance characteristics of each algorithm. PAES consistently exhibits robust performance, particularly at higher mutation probabilities, while SPEA-II and NSGA-II show varying levels of effectiveness dependent on the mutation rates and generation settings. The detailed examination of these algorithms under different configurations provides a deeper understanding of their operational strengths and limitations, preparing the groundwork for strategic application in multi-objective optimization scenarios. This comprehensive evaluation ensures that the selection of an appropriate algorithm can be tailored to specific optimization needs, enhancing decision-making processes in complex optimization environments.

7.2.5 Decision Making

Transitioning from a technical analysis of MOEA performance, the next section delves into the application of these results within a business context, aiming to identify the

most advantageous outcomes. The approach begins by narrowing down the pool of resources close to a target level of ten. Subsequently, the attention shifts to maximizing the output, prioritizing solutions that offer the highest productivity given the approximate resource amounts. Lastly, the analysis considers lead time, with a preference for solutions that tend to generate lower lead time. By applying these criteria, the section will distill critical insights from the MOEA results, aligning technical optimization with business objectives to support informed decision-making.

Implementing a resource filter of 10 (plus or minus 1) on the aggregated results yields 434 entries, among which 16 are non-dominated. These selected results exhibit a lead time ranging from 7,158 seconds to 14,259 seconds and output levels between 123 and 194. In pursuit of achieving higher delivery outputs, a subsequent filter targets results exceeding an output of 150. This refined filter reduces the pool to 29 results, with 7 maintaining their non-dominated status. These final 7 solutions are reflected in Table 28.

Table 28: Final Results of MOO

Results	Lead Time (s)	Output	Total Resources
1	11,611	171	10
2	13,876	179.6	9
3	12,649	183	10.5
4	12,228	180.4	10.5
5	10,560	151	10
6	11,270	193	11
7	14,258	195	11

Through an evaluation of these points, the minimum lead time is chosen as the target, and the final choice of the MOO. The seven data points that form the Pareto

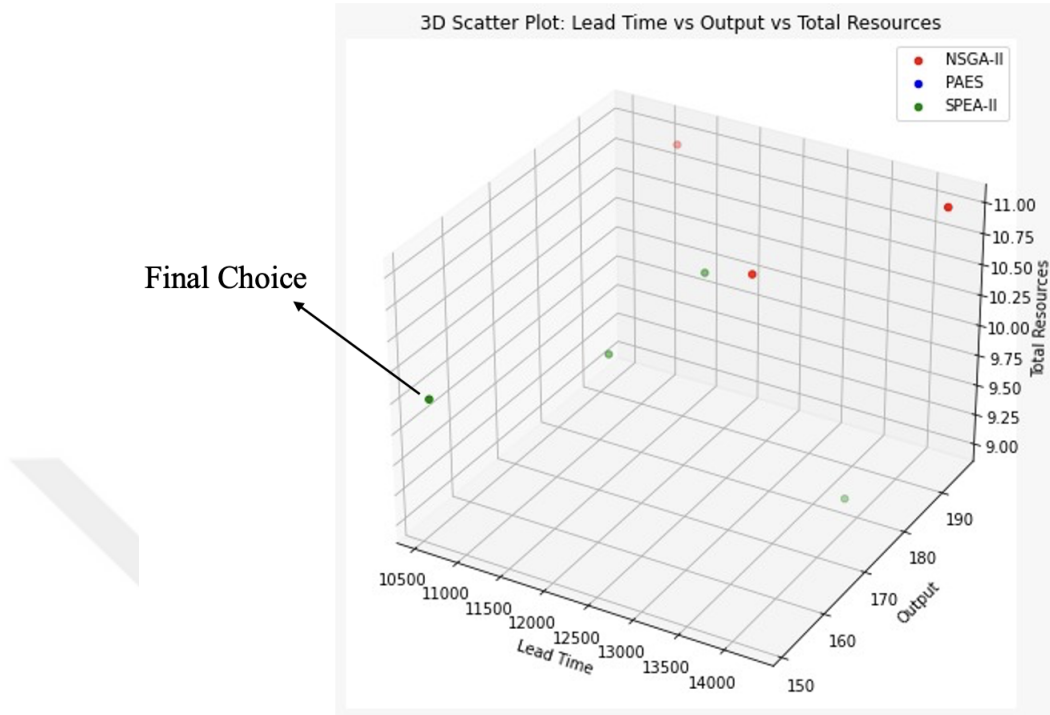


Figure 20: Pareto Front of Final Results and Final Choice

Table 29: Feature Set of Final Choice

Analysis	Development	Test	WIP	Priorities
3.5	4	2.5	15	p1:1 p2:3 p3:2

front and the final selection of the MOO are graphically represented in Figure 20.

The optimal result features an average lead time of 10,560 seconds and 151 completed work items, produced by a total of 10 resources. The configuration of the features for this choice is outlined in Table 29.

Reflecting on earlier analyses (refer to Table 21), it was noted that all machine learning algorithms had previously picked a WIP of 27 with priority set of 2, 3, and 1 to maximize output. The findings from the final MOO results suggest that a WIP level of 15, combined with the reordered priorities of 1, 3, and 2, significantly enhances lead time, offering a strategic insight into effective resource and priority management.

This final insight, also proves the importance and impact of WIP at lead time.

7.3 Discussion

The exploration of multi-objective optimization algorithms began with a detailed comparative analysis of SPEA-II, PAES, and NSGA-II across varying parameters. Through conducting runs with different number of generations and mutation probabilities, strengths and limitations of each method are highlighted. Overall, it was observed that PAES outperformed the other two algorithms in this setting.

Transitioning from a purely technical evaluation, the analysis was then steered towards incorporating business decision-making perspectives. By implementing filters based on resource allocation and output objectives, the study refined over 37 million potential solutions down to a focused set of seven candidates forming a Pareto front. After an observation of the last set of solutions, the final choice is made based on the business priorities. The final choice brings the key principles of Kanban to the table, and proves the importance of applying a WIP limit. This elicitation provides a decision-making process consisting of trade-offs, and shows effectiveness of MOEAs in managing complex decision spaces.

The insights derived from this analysis not only advance the theoretical framework of resource optimization but also offer practical guidelines for managers in dynamically adjusting their Kanban systems to achieve desired outcomes effectively.

CHAPTER VIII

CONCLUSION

This dissertation provides a comprehensive investigation into the optimization of software development teams working with Kanban. Given the continuous trade-offs in resource allocation, task management, and WIP limits within software development, the study adopts a holistic approach combining an initial solution phase, an extended phase, and an exploration of multi-objective optimization. As a whole, the research leverages simulation models, and advanced analytical techniques to provide diverse perspectives effectively.

The initial phase begins with a detailed development of a simulation model that reflects the real-world operational complexities of software development teams working with Kanban. The simulations form the underpinning factor of the study and the data generated by the simulations are used in all sections. The initial simulation model processes three types of work with priorities through three stations under WIP limits by various resources to generate insights about output, lead time, and efficiency. The novelty of the simulation model is the blocking scenario which represents a work item being blocked instead of a typical machine breakdown scenario in manufacturing. The simulations are integrated into a heuristic, a two-step, and a decision tree clearing function model to find an optimum resource allocation that maximizes output. The comparison shows that decision tree clearing function performs better than the heuristic and the two-step models.

The second phase extends the simulation model by integrating seven resource capabilities that can process three types of work at each station, so that one resource may handle multiple types of work sequentially at the same time interval. This

extension directly leads to a prioritization challenge and the introduction of priorities according to work types. This feature set increases the simulation horizon from 1,048,576 to 37,461,504. The extensive growth of the total possible combinations makes the solution approaches of the initial phase obsolete, and thus, ML techniques are used to investigate the large data universe. A 5000-large simulation data set is used to train Random forest, XGboost, and neural network to predict the vast simulation combinations. The numerical analysis shows that XGBoost outperforms the other two ML algorithms in the accuracy of predictions, and ML algorithms surface effective feature combinations that cannot be identified without using them.

The explorative section of the study takes a holistic view and accounts for lead time together with output and cost. The integration of lead time creates a multi-objective optimization problem, and it is addressed by three proven evolutionary algorithms: SPEA-II, PAES and NSGA-II built on output and lead time predictions generated by XGBoost. The comparative method tries various numbers of generations and mutation probabilities, and finds that PAES dominates SPEA-II and NSGA-II more than the others. The MOO concludes with a business perspective providing priorities of first number of resources, than output, and finally lead time. The business choices enable progress in finding the optimal number, and an optimal solution is achieved. The optimal solution demonstrates that WIP limits play a critical role together with the priorities in finding a balanced resource distribution. MOO section successfully shows how various optimization objectives can be balanced against each other to achieve the most efficient outcomes without disproportionately favoring one at the expense of others.

Critical contributions to the literature include:

- Introduction of a comprehensive simulation model: The thesis develops a simulation model that demonstrates a blocking mechanism on the blocked items

unlike the machine breakdown used in manufacturing. Moreover, the simulations integrate resource amount and capabilities, WIP limits, and priorities as parameters of software development teams working with Kanban.

- Decision tree clearing function applied to a resource allocation problem of a software development team working with Kanban: To find an optimal resource allocation, a heuristic and a two-step method were developed to benchmark the decision tree clearing function. As a result, the decision tree clearing function outperforms the benchmark models in cost scenarios and with mid-level data.
- Integration of capabilities and priorities into the simulation model and investigation of the combination universe with ML techniques: The extended simulation model creates a vast parameter universe that can only be estimated using ML techniques. Random forest, XGBoost, and NN algorithms are used, and XGBoost proved to be the most accurate.
- MOO and a comparison of three evolutionary algorithms: Introducing lead time as a third target in addition to output and cost creates a MOO perspective. Multiple objectives are explored through SPEA-II, PAES, and NSGA-II, with PAES providing the most results that dominate the others.
- Impact of WIP limits to lead time: Although it may seem straightforward, the impact of lead time is also a critical finding of the MOO since the optimal choice is a result of a lower WIP limit than the maximum output parameter set.
- Practical implications and frameworks: The findings and methodologies developed in this thesis offer actionable frameworks that can be directly applied in the industry. These frameworks help software development teams to not only understand their current operational inefficiencies but also to implement changes that lead to measurable improvements.

In conclusion, this dissertation significantly advances the literature among applications of Kanban in software development teams by providing deep insights into the optimization of resources. It offers both theoretical and practical contributions that bridge the gap between Kanban methodology and operational excellence in software development teams. The frameworks and models introduced here are expected to inspire further research and practical implementations across various industries where Kanban and similar methodologies are employed.



REFERENCES

- [1] U. S. Karmarkar, “Capacity loading and release planning with work-in-progress (wip) and leadtimes,” *Journal of Manufacturing and Operations Management*, vol. 2, no. 105-123, p. 37, 1989.
- [2] C. Ertaban and E. Albey, “Simulation based resource optimization using a decision tree clearing function,” *IEEE Access*, pp. 1–1, 2024.
- [3] M. Andreessen, “Software is eating the world, in: Wall street journal v. 20.08. 2011.” <https://www.wsj.com/articles/SB10001424053111903480904576512250915629460>, 2011. [Online; accessed 20.05.2024].
- [4] K. Beck, M. Beedle, A. Van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, *et al.*, “Manifesto for agile software development.” <https://agilemanifesto.org>, 2001. [Online; accessed 20.05.2024].
- [5] S. Denning, *The age of agile: How smart companies are transforming the way work gets done*. Amacom, 2018.
- [6] H. Lei, F. Ganjeizadeh, P. K. Jayachandran, and P. Ozcan, “A statistical analysis of the effects of scrum and kanban on software development projects,” *Robotics and Computer-Integrated Manufacturing*, vol. 43, pp. 59–67, 2017.
- [7] D. J. Anderson, *Kanban: successful evolutionary change for your technology business*. Blue Hole Press, 2010.
- [8] T. Ohno, *Toyota production system: beyond large-scale production*. crc Press, 1988.
- [9] S. Shingo and A. P. Dillon, *A study of the Toyota production system: From an Industrial Engineering Viewpoint*. CRC Press, 1989.
- [10] R. J. Schonberger, “Applications of single-card and dual-card kanban,” *Interfaces*, vol. 13, no. 4, pp. 56–67, 1983.
- [11] H. L. Gantt, “A graphical daily balance in manufacture,” *Journal of Fluids Engineering*, vol. 24, pp. 1322–1331, 1903.
- [12] J. Kelley and M. R. Walker, “The origins of cpm: A personal history,” *PM network*, vol. 3, no. 2, pp. 7–22, 1989.
- [13] H. Takeuchi and I. Nonaka, “The new new product development game,” *Harvard business review*, vol. 64, no. 1, pp. 137–146, 1986.

- [14] K. Schwaber, “Scrum development process,” in *Business object design and implementation*, pp. 117–134, Springer, 1997.
- [15] K. Schwaber and J. Sutherland, “The scrum guide.” <https://scrumguides.org/>. [Online; accessed 20.05.2024].
- [16] E. Ersoy, E. Çallı, B. Erdoğan, S. Bağrıyanık, and H. Sözer, “A longitudinal case study on nexus transformation: Impact on productivity, quality, and motivation,” *Journal of Software: Evolution and Process*, vol. 36, no. 5, p. e2615, 2023.
- [17] S. of Agile, “16th annual state of agile report.” www.stateofagile.com. [Online; accessed 20.05.2024].
- [18] J. Howl, “Queueing systems,” *Automatica*, vol. 3, no. 3-4, pp. 231–244, 1966.
- [19] A. K. Erlang, “Sandsynlighedsregning og telefonsamtaler,” *Nyt tidsskrift for Matematik*, vol. 20, pp. 33–39, 1909.
- [20] D. G. Kendall, “Stochastic processes occurring in the theory of queues and their analysis by the method of the imbedded markov chain,” *The Annals of Mathematical Statistics*, vol. 24, no. 3, pp. 338–354, 1953.
- [21] D. Gaver Jr, “A waiting line with interrupted service, including priorities,” *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 24, no. 1, pp. 73–90, 1962.
- [22] J. R. Jackson, “Jobshop-like queueing systems,” *Management science*, vol. 10, no. 1, pp. 131–142, 1963.
- [23] L. Kleinrock, *Queueing Systems, Volume I: Theory*. John Wiley & Sons, 1975.
- [24] J. A. Buzacott, “Queueing models of kanban and mrp controlled production systems,” *Engineering costs and production economics*, vol. 17, no. 1-4, pp. 3–20, 1989.
- [25] B. J. Berkley, “A review of the kanban production control research literature,” *Production and operations management*, vol. 1, no. 4, pp. 393–411, 1992.
- [26] R. J. Boucherie, X. Chao, and M. Miyazawa, “Arrival first queueing networks with applications in kanban production systems,” *Performance Evaluation*, vol. 51, no. 2-4, pp. 83–102, 2003.
- [27] M. I. Kellner, R. J. Madachy, and D. M. Raffo, “Software process simulation modeling: why? what? how?,” *Journal of Systems and Software*, vol. 46, no. 2-3, pp. 91–105, 1999.
- [28] J. A. García-García, J. G. Enríquez, M. Ruiz, C. Arévalo, and A. Jiménez-Ramírez, “Software process simulation modeling: systematic literature review,” *Computer Standards & Interfaces*, vol. 70, p. 103425, 2020.

- [29] R. L. Glass, I. Vessey, and S. A. Conger, "Software tasks: Intellectual or clerical?," *Information & Management*, vol. 23, no. 4, pp. 183–191, 1992.
- [30] N. S. Barghouti and D. S. Rosenblum, "A case study in modeling a human-intensive, corporate software process," in *Proceedings of the Third International Conference on the Software Process. Applying the Software Process*, pp. 99–110, IEEE, 1994.
- [31] M. Melis, I. Turnu, A. Cau, and G. Concas, "Evaluating the impact of test-first programming and pair programming through software process simulation," *Software Process: Improvement and Practice*, vol. 11, no. 4, pp. 345–360, 2006.
- [32] I. Turnu, M. Melis, A. Cau, A. Setzu, G. Concas, and K. Mannaro, "Modeling and simulation of open source development using an agile practice," *Journal of Systems Architecture*, vol. 52, no. 11, pp. 610–618, 2006.
- [33] D. Anderson, G. Concas, M. I. Lunesu, and M. Marchesi, "Studying lean-kanban approach using software process simulation," in *Agile Processes in Software Engineering and Extreme Programming: 12th International Conference, XP 2011, Madrid, Spain, May 10-13, 2011. Proceedings 12*, pp. 12–26, Springer, 2011.
- [34] D. J. Anderson, G. Concas, M. I. Lunesu, M. Marchesi, and H. Zhang, "A comparative study of scrum and kanban approaches on a real case study using simulation," in *International Conference on Agile Software Development*, pp. 123–137, Springer, 2012.
- [35] L. D. Otero, G. Centeno, A. J. Ruiz-Torres, and C. E. Otero, "A systematic approach for resource allocation in software projects," *Computers & Industrial Engineering*, vol. 56, no. 4, pp. 1333–1339, 2009.
- [36] A. Tregubov and J. A. Lane, "Simulation of kanban-based scheduling for systems of systems: initial results," *Procedia Computer Science*, vol. 44, pp. 224–233, 2015.
- [37] W. Wysocki and C. Orłowski, "A multi-agent model for planning hybrid software processes," *Procedia computer science*, vol. 159, pp. 1688–1697, 2019.
- [38] E. Weflen, C. A. MacKenzie, and I. V. Rivero, "An influence diagram approach to automating lead time estimation in agile kanban project management," *Expert Systems with Applications*, vol. 187, p. 115866, 2022.
- [39] N. Damij and T. Damij, "An approach to optimizing kanban board workflow and shortening the project management plan," *IEEE Transactions on Engineering Management*, pp. 1–8, 2021.
- [40] Y.-F. Hung and R. C. Leachman, "A production planning methodology for semiconductor manufacturing based on iterative simulation and linear programming calculations," *IEEE Transactions on Semiconductor manufacturing*, vol. 9, no. 2, pp. 257–269, 1996.

- [41] A. Srinivasan, M. Carey, and T. Morton, “Resource pricing and aggregate scheduling in manufacturing systems,” Tech. Rep. 88-89, 58, Carnegie Mellon University, Tepper School of Business, 1988.
- [42] S. C. Graves, “A tactical planning model for a job shop,” *Operations Research*, vol. 34, no. 4, pp. 522–533, 1986.
- [43] J. Asmundsson, R. Rardin, and R. Uzsoy, “Tractable nonlinear capability models for production planning,” Oct. 20 2005. US Patent App. 11/041,908.
- [44] J. Pahl, S. Voß, and D. L. Woodruff, “Production planning with load dependent lead times,” *4OR*, vol. 3, no. 4, pp. 257–302, 2005.
- [45] B. Selçuk, J. C. Fransoo, and A. De Kok, “Work-in-process clearing in supply chain operations planning,” *Iie Transactions*, vol. 40, no. 3, pp. 206–220, 2008.
- [46] H. Missbauer and R. Uzsoy, “Optimization models of production planning problems,” in *Planning production and inventories in the extended enterprise*, pp. 437–507, Springer, 2011.
- [47] E. Albey and R. Uzsoy, “Lead time modeling in production planning,” in *2015 Winter Simulation Conference (WSC)*, pp. 1996–2007, IEEE, 2015.
- [48] E. Albey, Ü. Bilge, and R. Uzsoy, “An exploratory study of disaggregated clearing functions for production systems with multiple products,” *International Journal of Production Research*, vol. 52, no. 18, pp. 5301–5322, 2014.
- [49] D. F. Irdem, “Evaluation of clearing functions’ fitting methodology and performance for production planning models,” Master’s thesis, Ind. Eng., NC State Univ., NC, USA, 2009. [Online; accessed 20.05.2024].
- [50] A. L. Samuel, “Some studies in machine learning using the game of checkers,” *IBM Journal of research and development*, vol. 3, no. 3, pp. 210–229, 1959.
- [51] S. Shalev-Shwartz and S. Ben-David, *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.
- [52] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of machine learning*. MIT press, 2018.
- [53] R. Yang, X. Ouyang, Y. Chen, P. Townend, and J. Xu, “Intelligent resource scheduling at scale: a machine learning perspective,” in *2018 IEEE symposium on service-oriented system engineering (SOSE)*, pp. 132–141, IEEE, 2018.
- [54] S. K. Singh and A. Jukan, “Machine-learning-based prediction for resource (re) allocation in optical data center networks,” *Journal of Optical Communications and Networking*, vol. 10, no. 10, pp. D12–D28, 2018.

- [55] J.-B. Wang, J. Wang, Y. Wu, J.-Y. Wang, H. Zhu, M. Lin, and J. Wang, "A machine learning framework for resource allocation assisted by cloud computing," *IEEE Network*, vol. 32, no. 2, pp. 144–151, 2018.
- [56] P. Pospieszny, B. Czarnacka-Chrobot, and A. Kobylinski, "An effective approach for software project effort and duration estimation with machine learning algorithms," *Journal of Systems and Software*, vol. 137, pp. 184–196, 2018.
- [57] H. Song, I. Triguero, and E. Özcan, "A review on the self and dual interactions between machine learning and optimisation," *Progress in Artificial Intelligence*, vol. 8, no. 2, pp. 143–165, 2019.
- [58] R. Veeranjanyulu, S. Boopathi, R. K. Kumari, A. Vidyarthi, J. S. Isaac, and V. Jaiganesh, "Air quality improvement and optimisation using machine learning technique," in *2023 International Conference on Advances in Computing, Communication and Applied Informatics (ACCAI)*, pp. 1–6, IEEE, 2023.
- [59] L. Breiman, "Random forests," *Machine learning*, vol. 45, pp. 5–32, 2001.
- [60] R. E. Schapire and Y. Freund, *Foundations of machine learning*. Mit Press, 2012.
- [61] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of statistics*, vol. 29, pp. 1189–1232, 2001.
- [62] T. Chen, T. He, M. Benesty, V. Khotilovich, Y. Tang, H. Cho, K. Chen, R. Mitchell, I. Cano, T. Zhou, *et al.*, "Xgboost: extreme gradient boosting," *R package version 0.4-2*, vol. 1, no. 4, pp. 1–4, 2015.
- [63] D. Nielsen, "Tree boosting with xgboost-why does xgboost win" every" machine learning competition?," Master's thesis, Dept. Inf. and Elect. Tech., Norwegian Univ. of Science and Tech., Norway, 2016. [Online; accessed 16.04.2023].
- [64] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain.," *Psychological review*, vol. 65, no. 6, p. 386, 1958.
- [65] K. Deb, K. Sindhya, and J. Hakanen, "Multi-objective optimization," in *Decision sciences*, pp. 161–200, CRC Press, 2016.
- [66] N. Gunantara, "A review of multi-objective optimization: Methods and its applications," *Cogent Engineering*, vol. 5, no. 1, p. 1502242, 2018.
- [67] V. Pareto, *Cours d'Economie Politique Professe a l'Universite de Lausanne*. No. Bd. 1, F. Rouge, 1896.
- [68] C. M. Fonseca and P. J. Fleming, "An overview of evolutionary algorithms in multiobjective optimization," *Evolutionary computation*, vol. 3, no. 1, pp. 1–16, 1995.
- [69] K. Deb, *Multi-objective optimization using evolutionary algorithms*, vol. 16. John Wiley & Sons, 2001.

- [70] O. L. De Weck, "Multiobjective optimization: History and promise," in *Invited Keynote Paper, GL2-2, The Third China-Japan-Korea Joint Symposium on Optimization of Structural and Mechanical Systems, Kanazawa, Japan*, vol. 2, p. 34, 2004.
- [71] J. D. Schaffer, "Multiple objective optimization with vector evaluated genetic algorithms," in *Proceedings of the first international conference on genetic algorithms and their applications*, pp. 93–100, Psychology Press, 2014.
- [72] F. Kursawe, "A variant of evolution strategies for vector optimization," in *International conference on parallel problem solving from nature*, pp. 193–197, Springer, 1990.
- [73] A. Zhou, B.-Y. Qu, H. Li, S.-Z. Zhao, P. N. Suganthan, and Q. Zhang, "Multi-objective evolutionary algorithms: A survey of the state of the art," *Swarm and evolutionary computation*, vol. 1, no. 1, pp. 32–49, 2011.
- [74] C. M. Fonseca, P. J. Fleming, *et al.*, "Genetic algorithms for multiobjective optimization: formulation discussion and generalization," in *Icga*, vol. 93, pp. 416–423, Citeseer, 1993.
- [75] E. Zitzler, M. Laumanns, and L. Thiele, "Spea2: Improving the strength pareto evolutionary algorithm," tech. rep., ETH Zurich, 2001. [Online; accessed 20.05.2024].
- [76] J. D. Knowles and D. W. Corne, "Approximating the nondominated front using the pareto archived evolution strategy," *Evolutionary computation*, vol. 8, no. 2, pp. 149–172, 2000.
- [77] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.
- [78] V. Baradaran, "Muti-objective optimization and simulation model to design the withdrawal kanban systems," *Research in Production and Operations Management*, vol. 9, no. 1, pp. 181–203, 2018.
- [79] S. Elattar, H. G. Mohamed, and S. A. Hussien, "A multi-objective optimization of secure pull manufacturing systems," *Applied Sciences*, vol. 12, no. 12, p. 5937, 2022.
- [80] S. D. Prasad, C. Rajendran, and O. K. Chetty, "A genetic algorithmic approach to multi-objective scheduling in a kanban-controlled flowshop with intermediate buffer and transport constraints," *The International Journal of Advanced Manufacturing Technology*, vol. 29, pp. 564–576, 2006.
- [81] N. B. Ruparelia, "Software development lifecycle models," *ACM SIGSOFT Software Engineering Notes*, vol. 35, no. 3, pp. 8–13, 2010.

- [82] H. D. Benington, “Production of large computer programs,” in *In Proceedings ONR Symposium on Advanced Programming Methods for Digital Computers*, pp. 15–27, IEEE, 1956.
- [83] K. Brennan *et al.*, *A Guide to the Business Analysis Body of Knowledge*. Iiba, 2009.
- [84] M. Choetkiertikul, H. K. Dam, T. Tran, T. Pham, A. Ghose, and T. Menzies, “A deep learning model for estimating story points,” *IEEE Transactions on Software Engineering*, vol. 45, no. 7, pp. 637–656, 2018.
- [85] “Overview kernel description.” <https://simpy.readthedocs.io/en/4.0.1/>. Accessed: 2022-08-30.
- [86] T. E. Oliphant, “NumPy: A guide to NumPy,” 2006–. [Online; accessed 16.04.2024].
- [87] IBM, *IBM ILOG CPLEX Optimization Studio*, 20.1 ed., 2021. Available at <https://www.ibm.com/analytics/cplex-optimizer>.
- [88] M. Droettboom, J. D. Hunter, *et al.*, “Matplotlib,” 2007–. [Online; accessed 16.04.2024].
- [89] R. Automation, “Arena simulation software,” 2021. Available at: <https://www.arenasimulation.com/>.
- [90] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
- [91] F. Chollet *et al.*, “Keras.” <https://github.com/fchollet/keras>, 2015. [Online; accessed 20.05.2024].
- [92] M. Antkiewicz and P. B. Myszkowski, “Balancing pareto front exploration of non-dominated tournament genetic algorithm (b-ntga) in solving multi-objective np-hard problems with constraints,” *Information Sciences*, vol. 667, p. 120400, 2024.
- [93] F. De Rainville, F. Fortin, M. Gardner, M. Parizeau, and C. Gagné, “Distributed evolutionary algorithms in python (deap),” *Journal of Machine Learning Research*, vol. 13, pp. 2171–2175, 2012.

VITA

Cihangir Ertaban completed his undergraduate degree in Management Engineering at Istanbul Technical University in 2007. He went on with his Master's degree in Industrial Engineering at Boğazici University between 2008-2011. He started his PhD studies in Industrial Engineering at Özyeğin University in 2017 while working at his full-time job. In his professional career, he worked at multiple organizations including Denizbank, Experteam and Turkcell in Türkiye. He moved to Germany in 2019. Currently, he is working as a Senior Manager at Accenture in Berlin. His interests include agile and lean ways of working. During his projects in agility, he decided to explore Kanban, and software development teams working with Kanban which led to his dissertation. His publications include the fields of performance management in agile teams, simulations, machine learning techniques and Kanban.