THE EFFECT OF HUMAN FACTORS ON THE SUCCESS OF SOFTWARE PROJECTS IN TURKEY

A Thesis

by

Nurver Şeyma Sel

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

Master of Science

in the
Department of Computer Science

Özyeğin University
October 2024

THE EFFECT OF HUMAN FACTORS ON THE SUCCESS OF SOFTWARE PROJECTS IN TURKEY

Approved by:

Asst. Prof. Yaşar Safkan, Advisor
Dept. of Computer Science
Özyeğin University

Prof. Hasan Sözer
Dept. of Computer Science
Özyeğin University

Prof. Emin Erkan Korkmaz
Dept. of Computer Eng.
Yeditepe University

Date Approved: August 13, 2024

To my lovely, supportive family and friends...

ABSTRACT

Software development is not a smooth process. Many software projects are delayed, failed, abandoned, or rejected for various reasons. One significant cause of failure is the neglect of human factors. Contrary to popular belief, software development is more of a human activity than a purely technical process, involving interactions between customers and development teams. The Agile Manifesto was a major step in acknowledging the importance of people in the software development process.

The formation of a cohesive software team underscores the relevance of human factors. Teams make decisions on many critical aspects of projects. Human interaction within the team is essential for building motivation, trust, communication, and mutual support, which are key to sound decision-making. To meet customer and stakeholder expectations, it is important to follow project delivery strategies that involve them throughout the process. Additionally, team managers are responsible for both the team and the project. Thus, both managers and teams must remain aware of human factors to ensure project success.

This thesis proposes a research model consisting of five human factors and their components. Their relative weight and importance in software project success are analyzed through surveys conducted with software development team members in Turkey. Two predictive models—Multiple Linear Regression (MLR) and Ridge Regression—are applied to the survey data. The MLR model is preferred due to the higher score R-squared (R^2). The MLR model suggests that "Project" and "Management" are the main factors contributing to software project success, followed by "Team." A total of 312 respondents participated in the survey, with 141 completed responses covering 63 questions.

ÖZETÇE

Yazılım geliştirme süreci sorunsuz bir süreç değildir. Bunun kanıtı olarak, birçok yazılım projesi çeşitli nedenlerle hâlâ gecikmekte, başarısız olmakta, terk edilmekte veya reddedilmektedir. İnsan faktörlerinin göz ardı edilmesi, yazılım projelerinin başarısız olmasının başlıca nedenlerinden biridir. Sanılanın aksine, yazılım geliştirme müşteriler ve yazılım geliştirme ekipleri arasında gerçekleşen tamamen teknik bir süreçten daha çok bir insan faaliyetidir. Agile Manifestosu, yazılım geliştirme sürecinde insan faktörlerinin önemini tanımada önemli bir adım olarak görülebilir.

Uyumlu bir yazılım ekibinin oluşturulması, insan faktörlerinin önemini vurgular. Ekipler, projelerin birçok kritik aşamasında önemli kararlar alırlar. Ekip içindeki insan etkileşimi, motivasyon, güven, iletişim ve karşılıklı destek oluşturmak için hayati öneme sahiptir ve bunlar, sağlıklı karar alma süreçlerinin anahtarıdır. Müşteri ve paydaş beklentilerini karşılamak için, onları süreç boyunca projeye dahil eden teslimat stratejilerini izlemek de önemlidir. Ayrıca, takım yöneticileri, ekip ve proje için önemli sorumluluklar taşımaktadır. Bu nedenle, hem yöneticilerin hem de ekiplerin proje başarısı için insan faktörlerinin farkında olmaları gerekmektedir.

Bu tez, beş insan faktöründen ve onların bileşenlerinden oluşan bir araştırma modeli önermektedir. Bu faktörlerin yazılım projesi başarısındaki göreceli ağırlığı ve önemi, Türkiye'deki yazılım geliştirme ekipleriyle yapılan anketler aracılığıyla analiz edilmiştir. Anket verilerine, Çoklu Doğrusal Regresyon (MLR) ve Ridge Regresyon olmak üzere iki tahmin modeli uygulanmıştır. MLR modeli, daha yüksek R-kare (R^2) puanı nedeniyle tercih edilmektedir. MLR modeli, yazılım proje başarısına en çok katkı sağlayan faktörlerin "Proje" ve "Yönetim" olduğunu, ardından "Takım" faktörünün geldiğini öne sürmektedir. Ankete 312 kişi katılmış, 141 tamamlanmış

yanıt alınmış ve bu yanıtlar 63 soruyu kapsamaktadır.

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to all those who have supported me throughout the journey of completing this thesis.

First and foremost, I would like to thank my advisor, Asst. Prof. Yaşar Safkan, for their invaluable guidance, support, and insightful feedback. Your consideration of my thoughts was very valuable to me. Since our first phone call, I have felt lucky about your impact on my life.

I would like to express my gratitude to the professors at Özyeğin University for their valuable knowledge, shared experiences, and vision. Thanks to them, I feel more confident and well-equipped in my field as I leave this university.

I am also immensely grateful to the jury members, Prof. Hasan Sözer and Prof. Emin Erkan Korkmaz, for their time, effort, and constructive comments.

A heartfelt thank you to my dearest friends (Hazal, Aleyna, Şima, Gamze, Selin, Yasemin, Yeşim, Damla, Afra, Sena O., Sena, and Zehra) for their constant encouragement and presence in my life, which made this journey enjoyable and supportive. I am grateful for the beautiful memories you have brought into my life. I am very lucky to have the strength and trust to lift each other up whenever we fall. I would also like to congratulate my dear friend Aleyna for walking the same paths. I am proud of us for the journey we have succeeded. Also, I want to congratulate my dear friend Hazal, on the new chapter of her life in London. My lovely friend Elif, thank you for your endless support and love, whether you are just a street away or on the other side of the world. May we have many more years of witnessing each other's success and life.

I am deeply indebted to my family for their unwavering love and support. To my

mom and dad, thank you for your sacrifices and love, and for instilling in me the value of education. To my lovely little brother, Ahmet, your belief in me has strengthened and motivated me. I owe my family where I am today and what I succeed in, love you. I want to thank my lovely Zeynep and Hüsna for growing up together and for the support we have given each other. I always feel grateful and lucky that we are always together.

Last but not least, I wanna thank me. This year has not been easy, but I succeeded in pushing through. As I reach the end of this journey, I am proud of myself for the memories and experiences. I eagerly and hopefully await what the future holds for me. With the wish that it includes my family, loved ones, happiness, and health...

Şeyma

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Numerous software products are integral to our daily lives. They are developed by teams through many stages, during which software development teams usually encounter various changes and challenges. Despite these efforts, there is always a possibility that a project might fail to reach completion. Projects can be rejected, delayed, failed, or abandoned for various reasons. They contribute to the high failure rate observed in software projects. This results in a waste of time, labor, and financial resources.

Critical success factors (CSFs) are defined as "for any business, the limited number of areas in which results if they are satisfactory, will ensure successful competitive performance for the organization." [1]. In other words, CSFs can be considered the areas in the organization must focus on achieving successful performance. For example, customer involvement is cited as one of the critical success factors for software projects in some research [2, 3], especially since it has been given the highest priority in the Agile Manifesto [4].

One of the primary causes of software project failure is a lack of understanding of CSFs in software projects, especially human factors. Some researchers draw attention to this deficiency: according to Nasir and Sahibuddin's research, CSFs are dominated by people factors [5]. Ahimbisibwe et al. also highlight that software projects often fail due to human factors rather than technical ones [6].

This thesis is motivated by the need to research critical success factors in software projects, focusing on human factors in Turkey through survey research. This study investigates how team dynamics, management, project strategies, preferred software

methodologies, and customer interactions impact software project success. Thus, the research model comprises these human-related factors to evaluate their impact on the success of software projects.

Teams and managers face limitations in terms of resources and time. However, by prioritizing critical parts of the project and allocating resources and time accordingly, they can significantly improve the outcomes of software projects. This strategic approach reduces waste of time, labor, and financial resources.

1.1 Software Development Between 1950-2000

The software industry came to be in the mid-1950s and has been in almost constant development and growth ever since. Software development can be defined as the process of designing, creating, testing, and maintaining computer software. It has evolved through many changes and stages to reach its current state today.

In the 1950s, the software industry was mainly for developing projects for governments [7]. Semi-Automated Ground Environment (SAGE) is one of the examples of such projects, which was developed for the U.S. and Canada. It is an air defense system designed to detect and prevent potential air strikes from enemies [8]. In addition, the SAGE system was the first extensive computer project developed between 1942 and 1962. According to estimates, there were 1,200 programmers in the United States at that time. On the other hand, out of the 2,100 total employees in Software Development Corporation, 700 were programmers. The Software Development Corporation was one of the pioneer software companies. In fact, it grew out of the Rand Corporation, which consisted of engineers working on the SAGE system. After that, this group of engineers separated and officially became Software Development Corporation in 1956 [9]. The SAGE program served as a university for programmers. It can be argued that the SAGE program has played a critical role in the history of software development [7].

According to Boehm, when he entered the software field in 1955 at General Dynamics, the approach was to, "Engineer software like you engineer hardware" [8]. He mentioned that everyone at General Dynamics was either a hardware engineer or a mathematician. They developed software for supporting aircraft or rocket engineering. Therefore, he referred to the 1950s as a time when "Software Engineering was like Hardware Engineering".

The distinction between software and hardware became more noticeable in the 1960s. Software modification was significantly easier than hardware modification, which led to the new approach of "code first, and then fix or make new adjustments until the user is satisfied." This "code and fix" methodology was suitable for its time [10]. However, it proved inadequate as software projects became increasingly complex over time.

The term "Software Engineering" was coined at the 1968 NATO Conference by Friedrich Bauer. Informally, Margaret Hamilton was the first to use the term "Software Engineering." She also worked on the SAGE project. Subsequently, she assumed the lead developer position for Skylab and Apollo, as shown in Figure 1. According to an unpublished oral history, she used the term "Software Engineering" to describe her role in the project in 1963 or 1964. Thus, she distinguished between hardware and software engineering [11].

Figure 1: Margaret Hamilton poses with the Apollo guidance software [12]

The software industry initially received limited attention because of its limited market share in the totality of computer business. Following the appearance of personal computers in the 1970s, the software industry grew remarkably quickly [7].

As a result, new approaches emerged to adapt to changing circumstances in software development. The "Waterfall" approach, which was popular in the 1970s, follows a sequential process that includes several consecutive phases. Each step is expected to be completed in sequence. The prescription is sequential, and while backtracking to previous steps *is* possible, it is discouraged and said to incur significant additional costs. This is implied in the name – while it is possible to go back up a waterfall, its natural direction is always "down" [13]. However, the Waterfall approach was costly and involved extensive documentation. According to the survey results conducted in many organizations, software cost exceeded hardware cost [14]. This outcome set the stage for trends for the next decade.

In the 1980s, there were no new approaches introduced in software development. Instead, there was an emphasis on improving productivity in existing methodologies

and finding solutions for scalability problems in software development [8].

In the 1990s, different software development approaches emerged with the increasing popularity of the Internet and the World Wide Web (WWW). Open Source Software Development is based on a collaborative approach that allows other programmers to access licensed source code, and they can fix, analyze, and make changes to them [15].

The Lean is other of the software development approach in the 1990s. Initially, Lean, as a manufacturing approach, originated from Japanese manufacturers and, in particular as a result of innovations at Toyota Motor Corporation [10]. The seven main bases of Lean manufacturing is optimizing the whole, eliminating waste, building quality, learning constantly, delivering fast, engaging everyone, and keep getting better [16]. Subsequently, its principles were applied to software development.

Throughout the years, various software development approaches have emerged. However, software development projects still have a high ratio of failure. A survey conducted by The Standish Group in 1994 in the U.S. revealed that only %16 of software projects were successful, based on data from several thousand IT projects [17].

Failed software projects not only result in the loss of time and labor but also have negative economic consequences. Billions of dollars are wasted annually due to failed software projects [18].

The pace of change in Information Technology (IT) is rapid. Therefore, heavyweight software approaches are becoming inadequate. The software development process has no single, exact cause of failure. Failures can occur in many different ways and under various circumstances.

Behind the software industry's failures are some technical problems, business decisions, project management, and human factors [18]. Software development should

not be considered solely as a technical process. It is made by a software team consisting of people with different roles. They develop software based on the customers' and the users' requests. In essence, it is made by people and for people.

Thus, in 2001, the Agile Manifesto was released by seventeen people who were aware of the high failure rate and the importance of human factors in software development [4]. They sought to find a solution to the problem in this industry, particularly by embracing changes in software development. Thus, the Agile Manifesto brought a different perspective to the history of software development.

1.2 Agile Software Development

Today, the world is changing rapidly. Expecting that software development has remained stable is not reliable. Software development must be dynamic and open to quick changes, especially with developing technology and changing customer and user demands. Traditional software development models have become insufficient because they minimize risk instead of embracing them [6]. Therefore, there is a need for new perspectives to meet the needs and demands of the industry. Agile software development (ASD) has made the sector more dynamic and open to quick changes.

1.2.1 Agile Manifesto

According to the Cambridge Dictionary, agile means being able to deal with new situations or changes quickly and successfully.[19] The software development process comprises various major and minor changes throughout projects. Many of these changes also require quick and successful handling.

Plan-driven software development methods were inadequate to meet the problems that occurred during the project quickly and successfully. To handle and discuss problems and improvements in software development, people gathered.

As a result, the Agile Manifesto was published in February 2001 by a group of seventeen individuals at The Lodge at Snowbird Ski Resort in the Wasatch Mountains

of Utah [4].

The Agile Manifesto consists of 12 principles [4].

1. **Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.**

2. **Welcome changing requirements, even late in development. agile processes harness change for the customer's competitive advantage.**

3. **Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.**

4. **Business people and developers must work together daily throughout the project.**

5. **Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.**

6. **The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.**

7. **Working software is the primary measure of progress.**

8. **Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.**

9. **Continuous attention to technical excellence and good design enhances agility.**

10. **Simplicity – the art of maximizing the amount of work not done – is essential.**

11. **The best architectures, requirements, and designs emerge from self-organizing teams.**

12. **At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.**

There are four core values in the Agile Manifesto [4].

1. **Individuals and interactions over processes and tools**

   Individuals and interactions are crucial in software development, as software is developed by humans for humans. Therefore, agile practices prioritize people and interactions between humans rather than processes and tools.

2. **Working software over comprehensive documentation**

   The Agile Manifesto promotes limiting documentation to essential levels, as delivering testable project components more frequently enables faster feedback and allows the team to respond fast to changes and new requirements.

3. **Customer collaboration over contract negotiation**

   After the Agile Manifesto, customers have become more important in software development. Customers now play a greater role, collaborating directly with the team to provide insights and feedback. Customer satisfaction is critical beyond just meeting contract requirements. Collaboration allows the team to incorporate customer ideas during the project.

4. **Responding to change over following a plan**

   Agile methodology emphasizes the importance of promptly responding to change, as software project plans are often unstable and subject to changes during development. Adhering strictly to the plan may be unsuitable, and flexibility to accommodate new requirements are crucial in the Agile software project lifecycle.

1.3 Agile Frameworks

1.3.1 Scrum

Scrum is one of the agile, lightweight frameworks developed by Jeff Sutherland and Ken Schwaber [20]. Scrum is widely used in the software industry; its popularity extends beyond software development. As a result, it has been adopted across various sectors.

Scrum is characterized by an iterative and incremental process. An iterative process means developing a product with repeated cycles, and this cyclic process is known as a Sprint in Scrum software development. Incremental development involves building the product in smaller, manageable parts. In Scrum, parts of the product are developed in each Sprint until the final product is delivered in the last Sprint. Thanks to these characteristics of Scrum, the team can deliver working subsystems of a final system frequently, providing feedback from stakeholders. Thus, Scrum embraces change and increases the potential for success.

1.3.1.1 Scrum Roles

The Product Owner, the Team, and the Scrum Master are three different roles in Scrum. The Product Owner is responsible for ensuring that the Product Backlog's most valuable functionality is developed first for each Sprint. The Product Backlog is a list of requirements. The Team is responsible for developing and delivering product functionality at the end of each Sprint. Teams are self-managing, self-organizing, and cross-functional, meaning they manage and decide their work within the Scrum framework. The responsibilities of the Scrum Master include ensuring all project members follow Scrum rules and practices.

1.3.1.2 Scrum Flow

All tasks are completed in Sprints. Each Sprint has a Sprint Planning meeting where the Team and the Product Owner discuss what will be done in the upcoming Sprint

[21]. The Product Owner presents the highest priority Product Backlog to the Team while the Team asks and discusses questions and also informs the Product Owner about their assessment of how much of the desired requirements it can deliver as functionality in the upcoming Sprint [21]. The Team conducts the Daily Scrum meeting to synchronize team activity and make a daily plan. Its duration is 15 minutes.

After finishing each Sprint, there are two meetings: Sprint Review and Sprint Retrospective. In the Sprint Review meeting, the Team presents what was developed during the Sprint to the participating stakeholders and the Product Owner. The Scrum Master conducts the Sprint Retrospective meeting with the team. It happens after the Sprint Review and before the next Sprint planning. The previous Sprint is evaluated, and improvements are identified for the next Sprint. This process continues until the project is completed.

1.3.2 Extreme Programming (XP)

Extreme Programming is one of the agile software methodologies. The "extreme" aspect of Extreme Programming comes from it advocating that "do your best and then deal with the consequences" which can make one feel exposed [22]. Its characteristic feature is a focus on technical aspects of software development. XP aims to help teams develop high-quality software and adapt to changing and evolving requirements. It embraces specific values, principles, and practices in software development, which are presented in table format and outlined sequentially (see Tables 1, 2, and 3).

Table 1: Values of Extreme Programming (XP)

Values	**Description**
Communication	Communication is crucial for forming a sense of team and efficient cooperation.
Simplicity	Simplicity eliminates unnecessary requirements and problems.
Feedback	Feedback contributes to identifying problems and making improvements to the project.
Courage	Courage discards failing solutions and encourages finding new ones.
Respect	Mutual respect is important for ensuring supportive and collaborative work environment.

Table 2: Principles of Extreme Programming (XP)

Principles	**Description**
Humanity	Software is developed by people.
Economics	Business value should be added to the project.
Mutual Benefit	Mutual benefits should be provided.
Self-Similarity	If something works in one situation, it can be tried elsewhere.
Improvement	Software development is imperfect, so always focus on improving processes.
Diversity	The team needs diversity because it also brings different ideas and solution about problems, even though diversity sometimes leads to problems and pitfalls.
Reflection	The team should also reflect on how and why they work.
Flow	Flow means delivering valuable software continuously instead of chunks.
Opportunity	Problems should be seen as an opportunity for change.
Redundancy	Problems can have different solutions, and each solution may not always solve the problem. Therefore, redundancy can be beneficial for preventing failures.
Failure	Failures can also comprise valuable information as they provide learning opportunities that can contribute to eventual success.
Quality	Quality should be pursued and not compromised. It is not a control variable.
Baby Steps	Many small steps can yield good results and make people feel safer.
Accepted Responsibility	Responsibility is accepted or taken, not given.

Table 3: Practices of Extreme Programming (XP)

Practices	**Description**
Sit Together	The team should develop the project in a big enough open space together.
Whole Team	The team should consist of individuals with diverse skill sets and perspectives essential for the project's success.
Informative Workspace	The workspace should represent the project and its current status.
Energized Work	The team should work as long as they can be productive and efficient.
Pair Programming	Two people should sit and work together on one computer.
Stories	The plan should be simplified for the customer's perspective.
Weekly Cycle	The team should plan work at the start of each week.
Quarterly Cycle	Work should be planned quarterly, and project progress and alignment with larger goals should be discussed.
Slack	The plan should include small tasks that can be omitted if necessary.
Ten-Minute Build	Automatically build the entire system and execute all tests in ten minutes.
Continuous Integration	Integration and changes should not exceed a few hours in the project.
Test-First Programming	The team should write a failing automated test before changing any code.
Incremental Design	The team should make small, safe improvements each day.

1.3.3 Kanban

Kanban originated in the Japanese manufacturing industry in the 1950s. The term "Kanban" means "signboard" in Japanese. Toyota used the Kanban in practice successfully. Kanban was brought into software development by David J. Anderson in 2004, during his management of a small, poorly performing IT team at Microsoft [23].

The Kanban board visualizes the assigned work of each team member, providing visibility to work for the team and highlighting priorities and bottlenecks [23]. Also, tasks are executed only if necessary, leading to the minimization of waste. The

Kanban approach limits work in progress (WIP) at each workflow stage and measures cycle time. This helps maintain team coordination, ensure a steady workflow, and decrease defects. These outcomes of Kanban foster good relationships and trust among the customers.

Kanban consists of five main principles: visualizing work, limiting work in progress, managing flow, making process policies explicit, and pursuing continuous improvement. Its popularity is not restricted to software development; it is also preferred in other industries.

1.4 Agile vs Traditional Software Development

Traditional and ASD are two predominant methodologies for developing and designing system software. These methods are different from each other in significant ways. These can be listed as various aspects of the development process, including development approach, flexibility, customer involvement, documentation style, team collaboration, risk management, and delivery strategy. The key differences and comparisons between these two predominant software development methodologies are presented in detail in Table 4.

Table 4: Comparison between agile and traditional software development

Agile software development	**Traditional software development**
1. The development approach is iterative and incremental, with regular feedback and adaptation.	1. The development approach is sequential, with distinct phases like planning, design, implementation, and testing.
2. It is flexible and can adapt to changes quickly.	2. Changes are difficult and costly to implement once the project is underway.
3. Constant customer involvement throughout the project.	3. Customer involvement is limited to specific milestones and reviews.
4. Documentation is minimal and just enough to support the development and use.	4. Documentation is comprehensive and detailed.
5. Teams are highly collaborative and cross-functional.	5. Teams are hierarchical, with clear roles and responsibilities.
6. There is a continuous risk assessment and mitigation in the project.	6. Risk assessment is made primarily at the beginning of the project.
7. There are frequent, incremental deliveries of working software.	7. There is a single, final delivery at the end of the project.

According to the study conducted by Tiwana and Keil on 720 software projects, using an inappropriate methodology is the most crucial risk factor contributing to project failure [24]. Therefore, understanding these differences can help organizations select the methodology that best suits their project goals and needs, which is important for increasing the chances of a project's success.

CHAPTER II

BACKGROUND

This chapter discusses software project failures and evaluation models. It examines historical examples, such as the Denver International Airport and Ariane-5 incidents, to illustrate the severe consequences of software project failures. The chapter then introduces various project evaluation models, including the Iron Triangle model and alternative models like Atkinson's Square Route, highlighting their strengths and limitations. It also discusses critical success factors and references Frederick P. Brooks Jr.'s "No Silver Bullet" to underscore the complexity of finding solutions to software development problems. Finally, the chapter emphasizes the importance of focusing on human factors in the software development process to achieve better project outcomes. Based on the systematic literature review, it has been decided to examine which human factors will be included in the research framework.

2.1 Failures and Challenges of Software Projects

Software projects can fail in various ways, resulting in delays, budget overruns, poor performance, or unsatisfied customers. This crisis in software delivery is often referred to as the 'Software Crisis [5].' Furthermore, software failures are not constrained by geographic region, industry, or market group or restricted to specific organizational size [25].

The Standish Group periodically publishes the 'CHAOS Report,' categorizing software project as successful, challenged, or failed, as shown in Table 5. The "Challenged" where even projects completed, they are invariably "over- budget, over the time estimate, and offer fewer features and functions than originally specified [25].

According to their research, many software projects face challenges in achieving success [26]. This also supports the notion that the software project process is inherently challenging.

Table 5: The Standish Group's Reports over time

Standish Group CHAOS Report 1994-2016			
Year	Succesful(%)	Challenged(%)	Failed(%)
2016	37	44	19
2015	36	45	19
2014	36	47	17
2013	41	40	19
2012	37	46	17
2011	39	39	22
2009	32	44	24
2006	35	46	19
2004	29	53	18
2000	28	49	23
1998	26	46	28
1996	27	33	40
1994	16	53	31

[27]

Many projects have encountered problems because of software issues. The Denver International Airport and the explosion of the European Space Agency rocket, Ariane-5, can be cited as notable extreme examples.

The opening of Denver International Airport was delayed over sixteen months due to major bugs in the baggage handling control software [28]. The cost caused by this delay exceeds the initial 100-million-dollar airport construction budget [29].

Ariane-5 exploded 40 seconds after launch on June 4th, 1996. The cost of this disaster was approximately $370 million [30]. According to the Inquiry Board Report (IBR), the underlying cause of this incident was software failure [31].

Performing research into reasons for software project failures is crucial as it is the primary method of improving software development practices and methodologies. As illustrated by these examples, abandoned, canceled, or rejected software projects lead to a waste of time, financial resources, and labor. Just as in any other sector of business life, experiencing failure is inevitable and part of the learning process. The results of such research are invaluable for teams, companies, and management.

Many researchers and individuals have discussed problems in software development. "No Silver Bullet" book was an impressive by Frederick P. Brooks Jr. on the topic. He mentions that "no silver bullet" or one-size-fits-all solution can guarantee the success of software projects [32]. Therefore, discussing software development's challenges and failures requires multiple approaches. In this thesis, the research model focuses on human factors that may provide solutions and insights into the software development process. As such, this topic should be researched in the context of current situations and updated as necessary. Furthermore, this topic should always remain open to new or different approaches. This is because every process involving humans and human factors should adapt, improve, and change according to human needs.

2.2 The Success of Software Projects

Success in software development is a hot topic for research studies. Achieving success in software development is often challenging. Software projects usually occur in a turbulent environment where countless problems and changes can arise throughout the project's life cycle. Numerous delayed, failed, abandoned, and rejected projects, stemming from various reasons, can indicate this situation. Researching successful software projects and their common features can provide a solid foundation for improving software project performance and for further research.

The Cambridge Dictionary defines success as 'the achieving of the results wanted or hoped for' [33]. However, the definition of success in software development is not so clear-cut.

First, there is a need for clarification regarding specific steps. How should success be evaluated, and based on which criteria? Additionally, who measures the success of projects? The meaning of success can differ from person to person. For example, delivering on time and meeting requirements may be the main indicators of the project's success for customers. However, this can be different for software development teams.

According to software teams, customer satisfaction or not exceeding the budget can be signs of project success. Furthermore, the evaluation of success can change within software development teams. For instance, project managers and software developers can have different perspectives on software project success. Therefore, some rules should be clarified for evaluation purposes.

2.2.1 Success Criteria

Many researchers have studied the definition of project success in software development. Identifying common success criteria can simplify the evaluation of projects.

There are different criteria for evaluating the success of software projects. To properly assess the performance of a software project, the evaluation should not be done from only one criterion. In other words, the evaluation should consider various criteria, which can also be combined.

Various models, from the past to the present, have been proposed to define success in projects. Initially, the Iron Triangle is used for project assessment in literature, and it is followed by Critical Success Factors (CSFs) [3]. Additionally, the Square Route model originated from some parts of the Iron Triangle [34].

2.2.2 The Iron Triangle

The Iron Triangle is a well-known success criteria model that has been preferred in project management for decades. Also called the Triple Constraint, it is used to how we infer success in projects [35]. The model consists of three components: cost, time, and quality. Since the 1970s, these three components of the Iron Triangle have been widely acknowledged as project success criteria [34]. For years, most project managers have preferred it as the standard way of defining project success. [36]. Due to the simplicity of the Iron Triangle, it can be considered one of the reasons for the enduring popularity of the model [35].

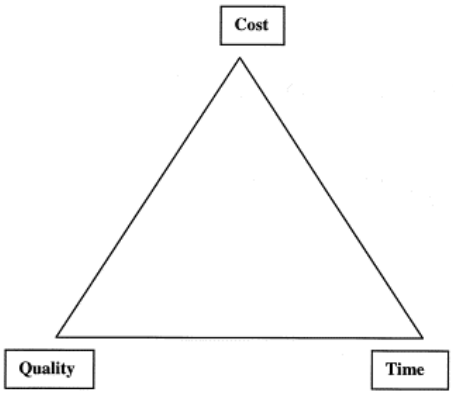

Figure 2: The Iron Triangle

2.2.2.1 Cost

Projects have an allocated budget; hence, managing the budget during the life cycle of the project is also significant. Delivering the project within the allocated or estimated cost can be a determinant of success in terms of cost criteria.

2.2.2.2 Time

Projects have a delivery deadline for the customer. Delivering the project to the customer within the specified time frame leads to project success in terms of the time criterion.

2.2.2.3 Quality

Due to the subjectivity of the quality, its definition can changed person by person. In general terms, delivering a good working product to the customer can be a determinant of success in terms of quality criteria.

The definitions of cost and time are clear; in other words, there is agreement on their meanings. However, quality is the most controversial component of the model [35]. This is because cost and time are objective, while quality is subjective. According to Atkinson, quality is a phenomenon that depends on different people's opinions and perceptions, which can often change throughout the project [34]. That

is why defining quality is not as simple as defining the other components of the model.

Within research studies on the Iron Triangle, there are some discussions specifically addressing the quality component. There are different ideas about this component. One idea is to divide the quality component into sub-parts. Turner and Huemann partitioned quality into product quality and process quality [37]. Wyngaard et al. suggest the idea that quality should be replaced with scope in the Iron Triangle [38]. Another approach to the quality component is adding an extra component to the Iron Triangle. Atkinson's "The Square Route" can be shown as an example of this approach [34]. The model details are mentioned under the 2.2.5 Section.

2.2.3 Limitations of the Iron Triangle

One of the main reasons for the popularity of the Iron Triangle is its simplicity. However, this simplicity of the model can pose problems, especially if it hinders the identification of real issues in projects.

As van der Hoorn and Whitty have discussed in their research, the Iron Triangle focuses on these three components rather than other complex parts of the project. Therefore, it can veil difficulties and challenges in projects, posing problems, especially for inexperienced individuals in project management [39].

According to Turner and Zolin, assessing the project in terms of time, cost, and quality is insufficient because success is not just completing the project's scope of work but also achieving business objectives. They mentioned that the Iron Triangle indicates project performance success but is not the best measure of project success [40].

Atkinson argues in his article that the reason for the continuous failure of project management can be old and ineffective criteria for measuring success [34]. He likens time, cost, and quality success criteria to the chance of matching two best guesses and a phenomenon. Therefore, there is a need to consider other factors in determining

project success, and he proposed a Square Route model for it [34].

According to De Wit, using the Iron Triangle to evaluate project management success instead of project success is more suitable [41].

Bronte-Stewart also deliberated on the limitations of the Iron Triangle, which ignores subjective and context-specific issues of projects and fails to consider important success criteria related to the overall impact and outcome of the project [42]. Therefore, project failure statistics may be misleading.

There is a consensus among some authors that the Iron Triangle is inadequate for determining project success since the model provides a limited view of the project perspective [41, 34, 39, 42, 40].

The reason for this is the need for a more comprehensive set of criteria to evaluate the project's success. In project success, customer satisfaction and the satisfaction of multiple stakeholders are crucial. Therefore, incorporating customer and stakeholder satisfaction could be a good way to broaden the project perspective.

2.2.4 Customer and Stakeholder Satisfaction

Since the 1970s, the significance of stakeholders' interaction has increased in projects [35]. This shift in project management practices acknowledges the significance of involving stakeholders throughout the project life cycle from the start to finish [43]. Furthermore, the definition of the project success perspective began to turn towards satisfying stakeholders' needs [44]. Subsequently, some papers mentioned that stakeholder satisfaction should also be considered for project performance assessment [34, 45, 40].

Turner and Zolin suggested that a more accurate and fair way of evaluating a project performance is to consider the views of multiple stakeholders across multiple time frames, which are the end of the project, plus months, and plus years [40].

Shenhar et al. proposed a multidimensional universal framework to assess project

success [45]. The model consists of four distinct success dimensions: Project efficiency, Impact on the customer, Business success, and Preparing for the future. Customer satisfaction is evaluated under the 'Impact on the Customer' dimension of the framework, along with meeting customer needs and solving customer problems. The remarkable outcome of their study indicates that customer satisfaction was the number-one criterion for achieving project success, followed by the components of the Iron Triangle [45].

Atkinson mentioned that customers and users are stakeholders and that stakeholder criteria should also be added to performance assessment. He included the benefits of the stakeholder community in his proposed new success framework, the Square Route [34].

2.2.5 The Square Route

The concept of the Iron Triangle has been widely used for evaluating projects over many years. In research, Atkinson questioned, "Why has project management been so reluctant to adopt other criteria in addition to the Iron Triangle?" [34]. In his analysis, he classified and defined errors into Type I and Type II within project contexts.

2.2.5.1 Type I Error

Type I errors occur when actions are executed incorrectly or inadequately, such as poor planning, inaccurate estimation, or insufficient control.

2.2.5.2 Type II Error

Type II errors occur when actions are forgotten or not executed well, such as using incomplete success criteria.

He contended that the Iron Triangle model represents a Type II error—it is not entirely incorrect but could be improved. Based on this perspective, Atkinson put

forward a more comprehensive framework for project assessment.

Thus, the Square Route includes the Iron Triangle with three additional components: The Information System, Benefits (Organizational), and Benefits (Stakeholder) [34]. The Information System component focuses on the importance of robust and reliable information systems in project management. The Benefits (Organizational) component highlights the organizational benefits derived from the project. It looks contribution of the project to the organization's broader goals and strategic objectives. The Benefits (Stakeholder) component emphasizes the importance of meeting the expectations and needs of various stakeholders.

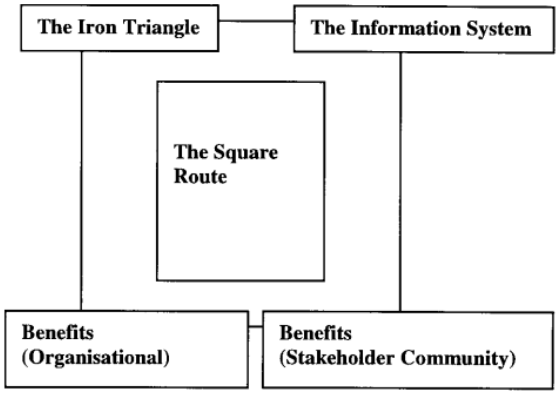

Figure 3: The Square Route

[34]

The Square Route provides a broader approach to project evaluation with these additional three components on the Iron Triangle.

2.2.6 Critical Success Factors

Critical Success Factors (CSFs) method was introduced in the article of "Chief Executives Define Their Own Data Needs" by John F. Rockart in May 1978. CSFs are "for any business the limited number of areas in which results if they are satisfactory, will ensure successful competitive performance for the organization"[46]. "They are

the few key areas where "things must go right" for the business to flourish" [1]. Put differently, if areas are good, the business will be successful; however, if areas are inadequate, problems and struggles may arise in business [1].

The motivation behind CSFs is to decide on specific areas crucial for project performance. This enables teams to concentrate their efforts on these specific areas. The rationale behind this approach is that managers and teams have limited resources and time. They can prioritize their resources in these areas; thus, they can make the difference between success and failure [47].

The method will certainly vary from manager to manager, company to company, and industry to industry. Therefore, an important point about CSFs is that they are not a standard set of measures or "key indicators" [47]. In another way, CSFs must be tailored and refined based on specific sectors.

CSFs are studied and researched in various areas, including the software development industry. Different studies about CSFs in software development exist, and their research results show that human-related factors also play a role in the success of software projects [5, 3, 2, 48, 49].

According to the extensive literature survey on critical success factors affecting software projects conducted by Nasir and Sahibuddin, twenty-six CSFs were found to be related to software project success [5]. They grouped twenty-six CSFs as seven people-related, sixteen process-related, and three technical-related. Only 6% are technical factors, and the rest are non-technical processes. The research results support the statement, "The success or failure of a project is seldom due to technical issues" [50].

The study by Chow and Cao aims to identify and provide valuable understanding of the CSFs that help in achieving success in software development implemented with agile methods [2]. The study uses data from 408 respondents across 109 agile projects in 25 countries. They conducted failure research across four dimensions:

Organizational, People, Process, and Technical. Additionally, success was researched across five dimensions, with an additional dimension added for the Project. They listed possible factors which may affect the success or failure of ASD projects. For the "People" dimension, possible failure, and success factors are listed in Tables 6 and 7.

Table 6: Possible failure factors

Dimension	Factors
People	Lack of necessary skill-set
	Lack of project management competence
	Lack of teamwork
	Resistance from groups or individuals
	Bad customer relationship

Table 7: Possible success factors

Dimension	Factors
People	Team members with high competence and expertise
	Team members with great motivation
	Managers knowledgeable in agile process
	Managers who have light-touch or adaptive management style
	Coherent, self-organizing teamwork
	Good customer relationship

After applying reliability and factor analyses, the final list consists of 12 possible CSFs. These were turned into 48 hypotheses, and each factor was evaluated across four project success categories: quality, scope, time, and cost. For the people dimension, they researched team capability and customer involvement. According to the results of the analyses, ten hypotheses were supported, while the remaining thirty-eight hypotheses were rejected. In supported hypotheses, %50 is technical, and %50

is non-technical factors. Table 8 represents the supported non-technical hypotheses along with their accepted success attributes. The primary contribution of this study is to reduce numerous anecdotal factors for success through the analysis of survey data. [2].

Table 8: Supported non-technical factors and four success attributes in the study of Chow and Cao

Non-Technical Factors	**Scope**	**Timeliness**	**Quality**	**Cost**
Process				
Good agile project management process			X	
Organizational				
An agile-friendly team environment			X	
People				
A strong customer involvement	X			
Team capability		X		X

[2]

In the research by Misra et al., 14 factors were hypothesized and categorized into two groups: People and Organizational factors [48]. According to the analysis of 241 survey respondents, nine of them were found to have a statistically significant relationship with "Success". These are customer satisfaction, customer collaboration, customer commitment, decision time, corporate culture, control, personal characteristics, societal culture, and training and learning. This research indicates that most of the proposed non-technical factors impact the success of software projects.

The study by Tam et al. researched the people factors that contribute to agile software project success [3]. The study is a combination of people factors found by Chow and Cao and Misra et al. [2, 48]. The proposed model includes Personal Characteristics, Societal Culture, Team Capability, Customer Involvement, and Training

& Learning. They found that team capability and customer involvement are influential factors. Additionally, team capability is affected by personal characteristics and societal culture.

There is a new article about four human factors: team capability, customer involvement, psychological safety, and team autonomy and their contributions to ASD success based on 177 responses from the survey [49]. They applied a Partial Least Squares (PLS) analysis to the research model. The study finds that customer involvement and team capabilities have a significant impact on the software project's success. Moreover, psychological safety is a significant indirect success factor. Team autonomy acts as a mediator that affects the indirect impact of psychological safety on project success.

The role of CSFs in project success or failure is significant [47]. When properly managed and addressed, these factors significantly enhance the likelihood of achieving project success [5]. Project managers should consistently focus on CSFs to improve project performance. Additionally, studies on CSFs emphasize the crucial role of people-related factors in the success of software projects, leading to extensive research on human factors through literature reviews.

CHAPTER III

HUMAN FACTORS

The software development process comprises both technical processes and components. In essence, people develop software for the benefit of people. Contrary to common expectations, the software development process is not solely focused on technical aspects; it relies heavily on human factors. Human factors significantly affect the success of software projects. Studies on CSFs also support this, indicating that human factors have a substantial impact on the success of software projects [2, 48, 5, 51, 52, 6, 3, 49]. Therefore, discussing human factors is crucial for understanding software project performance.

In the past, hardware was the most costly component of a system. Boehm's supervisor at General Dynamics remarked, "We are paying $600 an hour for the GD ERA 1103 computer and $2 an hour for you and I want you to act accordingly." on his first day [8]. However, this situation has since reversed. Today, people are the most expensive resource in software development [53]. This shift underscores the importance of discussing and researching human factors in the software development process.

This thesis research examines human factors in software development, specifically focusing on software development teams, preferred software methodologies, and project management, customers, and project-related aspects, along with their respective subsections.

Each section provides details of these human factors, supported by relevant literature, to demonstrate their significance in achieving successful software projects. Based on this information, the impact and importance of these human factors on

project success have been explained. Motivated by these findings, a new model has been developed. To statistically evaluate the model, related questions were included in a survey conducted in Turkey and analyzed in this thesis.

3.1 Software Development Teams

Software development is not an individual, intellectual endeavor [53]. Instead, it is a collaborative, social task comprising people with various roles and interactions in the process [54]. The software development team can be defined as a group of individuals who work collaboratively to design, create, test, and maintain computer software in complex organizational environments.

The team is a critical aspect of software development, as is any process involving human interaction. Software development would be impossible without teamwork, leaving nothing to discuss regarding the software project's success. Therefore, it is important to research the success factors of software project teams and teamwork. A software development team comprises various components and dynamics. This section discusses communication, communication overhead, personal characteristics, team environment, and team capability.

3.1.1 Communication

Communication is essential for human interactions. In a software project, when we refer to communication, it means the total communication among a group of people working on the same project, both to understand what each other is doing and to make collective decisions [55].

Communication is crucial in a software development team where individuals have diverse responsibilities and duties. It extends beyond internal team interactions to include communication with customers and stakeholders, making it a vital component of the entire software development process.

Agile methodologies advocate for small, collaborative teams working closely together. This structure promotes effective collaboration and communication among team members, facilitating adaptation to changes and quick delivery of valuable software. Agile management suggests that team sizes should ideally range from three to nine members[56], with team efficiency peaking within this range [57]. For instance, the ideal team size for Scrum is seven (plus two or minus two).

Communication is the most important element of agile methodology [58]. Software development teams using agile methodology have meetings to inform each other about development and changes in the project. These meetings are typically held daily, enhancing both formal and informal communication [59]. Furthermore, many agile principles prioritize face-to-face, informal, and daily communication over relying solely on documentation [59].

Pair programming can be a good example of the importance of communication in ASD. In pair programming, two programmers work together on the same algorithm, design, or code using one computer. There are two roles: driver and navigator. The driver writes the code while the navigator reviews it for issues and makes suggestions. Pair programming is a key component of the Extreme Programming (XP) methodology [22].

A survey was conducted among individuals who have experienced pair programming. According to the survey, good communication skills were ranked as the third most crucial characteristic of a pair programming partner and the first characteristic of pair programming teams [60]. Given the extensive communication involved in pair programming, this result is expected.

Agile teams demonstrate their best performance in a common space, often called the war room. Working in the same room primarily facilitates continuous interactive communication and supports team collaboration [52, 61]. This strategy has been shown to significantly enhance team productivity [62, 63].

One of the strengths of ASD lies in frequent communication, enabling teams to stay informed about evolving software demands and the team's overall progress [59]. They can notice the problem faster. Thus, agile software teams can act faster when encountering a problem or change throughout the project life cycle. Moreover, regular communication is important for establishing trust within the team environment [59].

According to Ancona and Caldwell, the size of the team directly impacts both performance and communication [64]. While effective communication can enhance a software project's performance, excessive communication can also negatively affect it due to communication overhead [65].

3.1.1.1 Communication Overhead

Communication, or miscommunication, is one of the root causes of project failure [52]. Contrary to popular belief, an excessive increase in the number of people in the software development team does not make the project progress faster or better; instead, it can cause the project to slow down or even fail. Numerous studies have explored the relationship between communication problems due to increasing team size and the success of software projects [66, 67, 52].

One of the common illusions in software projects is that when a project is late, just adding more people will make it finish sooner. Many projects still fall for this illusion, initially pointed out in "The Mythical Man-Month" by Fred Brooks in 1975 [66]. This notion not only fails to finish the project earlier but also causes it to be even more delayed. It overlooks the nature of software development, which is a collaborative, social task requiring extensive communication [53]. In essence, software development thrives on rich interactions among individuals.

Another issue is the wrong choice of using man-month as a unit for measuring the size of a job [66]. In Figure 4, this unit implies that men and months are interchangeable. This assumption holds true only if there is no communication in the project.

However, this condition is unrealistic because even the smallest project requires at least two people: the developer and the user. In this situation, communication is inevitable for the software process to understand the user's needs.

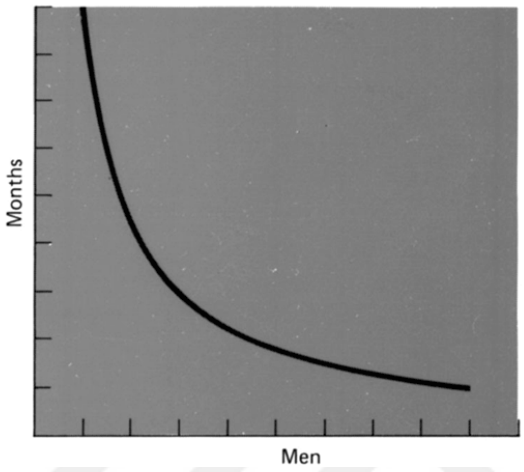

Figure 4: Time vs. number of workers—perfectly partitionable task [66]

Adding more individuals to a project can speed up until a specific limit is reached. Beyond this threshold, rather than finishing ahead of schedule, it may lead to the project being completed later than expected. In some cases, this can even lead to the project's failure or abandonment. Therefore, the actual situation of adding more people to the team can be observed in Figure 5.

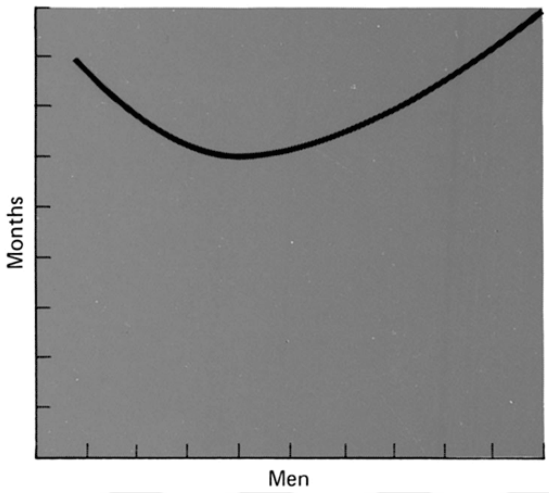

Figure 5: Time vs. number of workers—task with complex interrelationships [66]

Malone and Crowstone define coordination as managing dependencies between activities [67]. Increasing team size comes with difficulties, such as communication overhead, making it more challenging to keep the team informed about developments and ensure team coordination.

A study by Lalsing et al. examined the relationship between three different team sizes and project performance in agile teams using Scrum [52]. Three teams of different sizes were compared, and the results indicated that the larger team outperformed the smaller teams in most performance metrics. The research attributed this to the increased communication channels in the larger team.

Communication channels are calculated as the number of people multiplied by the number of people minus one and then divided by two. The formula for calculating communication channels is given by N, which represents the number of people in the team.

$$\frac{N \cdot (N-1)}{2} \tag{1}$$

For example, the communication channels among the three people are calculated

using the formula:

$$\frac{3 \cdot (3-1)}{2} = 3$$

It continues, as you can see in Figure 6.

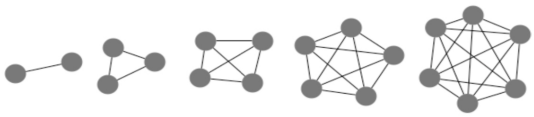

Figure 6: The communication channel

[52]

The formula indicates that communication channels grow exponentially as team size increases. In other words, increasing the team size and communication channels do not have a linear relationship.

Consequently, coordination and communication within the team become more challenging as team size increases. Thus, there is a trade-off between team size and communication. Deciding on the appropriate team size is crucial for project performance. Increasing the team size does not necessarily lead to better performance; it often results in increased communication overhead.

The observed result in these studies [66, 67, 52] is due to the increased complexity and potential for miscommunication as team size grows, which can negatively impact software project success. The increasing communication overhead makes effective communication and coordination more challenging. This situation can cause delays and various problems during the project life cycle. Communication is a vital component of software project success. Therefore, when forming a software team, the influence of communication on the success of software projects should be carefully considered.

3.1.2 Team Environment

The team environment is a fundamental pillar of successful teamwork, particularly in the context of software development projects. The Agile Manifesto, a cornerstone document in modern software development methodologies, highlights the importance of fostering a conducive environment for teams to succeed. One of its principles emphasizes that, "Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done." [4].

Trust positively influences the team's performance [68]. DeOrtentiis et al. also mentioned that trust strengthens bonds between team members, thereby turning the team into a cohesive unit that successfully achieves its goals. [68]. Akgün et al. found that trust among team members positively impacted the team potency during the project after examining 53 software development project teams in Turkey. [69]. Trust also positively affects knowledge-sharing. Sharing information and expertise is crucial for meeting customer requirements because the nature of the software development process is highly information-intensive [70].

The positive and negative impacts of the team environment on the success of software projects have been the subject of research by numerous studies. A study by Chow and Cao found a statistically significant relationship between an "agile-friendly team environment" and the "Quality" success attribute [2]. Conflicts in the team environment lead to instability within the team, negatively affecting the "Time" and "Budget" success attributes, as found statistically [71]. From different perspectives but leading to similar outcomes, there is a relationship between the team environment and software project success. To increase the chances of success in a project, teams should be provided with an appropriate team environment.

3.1.3 Personal Characteristic

The experience and competence of team members are crucial, but their personal attributes, such as honesty, collaboration, responsibility, readiness to learn and teamwork are equally important [72].

One of the ways to provide a suitable environment is by considering the personality characteristics of the people when assembling the team. The success factors research conducted by Misra et al. included the personal characteristics of team members. Their hypothesis, "The better the personal characteristics of the team members in a project, the more likely would be the success of the ASD projects," was accepted [48]. Furthermore, Tam et al. combined the studies of Misra et al. and Stankovic et al. However, their findings show that personal characteristics do not directly affect agile project success, contrary to Misra et al.[3]. Instead, personal characteristics significantly influence success through team capability and customer involvement, highlighting the importance of these mediators in agile projects.

3.1.4 Team Capability

Team capability refers to the reliable capacity of a project team to effectively utilize knowledge in challenging work environments [73]. Teams develop software products in these challenging environments. Various studies have examined team capability and its effect on the success of software development projects [2, 51, 49, 3]. Some research has found a likely relationship between project success and team capability.

First of all, according to Chow and Cao's research results, there is a statistically significant relationship between team capability and success attributes "Timeliness" and "Cost." [2]. In contrast, Stankovic et al. did not find a statistical relationship between team capability and success attributes: timeliness, cost, quality, and scope [51]. After that, studies by Barros et al. found that team capability strongly affects the success of ASD [49]. Using the PLS-SEM model, Tam et al. demonstrated that

team capability significantly explains the variance in the success of an ASD project [3].

These research findings align with Barry Boehm's emphasis on the importance of team capability over technical skills. Boehm highlights that a team's overall capability is more crucial than individual technical skills, such as knowledge of programming languages, proficiency with software tools, or even experience with specific applications [74]. This is because a highly capable team can better adapt to changing requirements, effectively collaborate to solve complex problems, and leverage collective intelligence to innovate and improve project outcomes.

3.2 Project Management

Project management is a critical discipline encompassing the "application of knowledge, skills, tools, and techniques to project activities to meet the project requirements" [75]. An analysis of 250 large software projects from 1995 to 2004 reveals an interesting pattern: unsuccessful projects struggled not due to technical personnel but due to challenges in project management [76]. Thus, effective project management is essential for the success of software development projects. This section discusses various aspects of project management and their impact on software project performance. Specifically, this thesis research focuses on the roles of the manager in project management.

3.2.1 Team Manager

Indeed, team managers play a crucial role in the software development process and are essential for achieving software project success.

Maintaining a harmonious and creative work environment is one of the key responsibilities of a team manager [77, 78]. Such an environment involves establishing trust, fostering motivation, and encouraging problem resolution from different perspectives [77]. According to Liang and Liu, providing direction for creating and managing

diverse teams can further enhance team performance by promoting a variety of viewpoints and innovative solutions [78]. The importance of the team environment on project success is discussed in section 3.1.2.

The team manager's provision of leadership and direction is crucial for the team to understand their tasks and responsibilities [77]. This also ensures coordination within the team. Furthermore, leadership support encourages the team to share problems, feedback, and challenges with each other, thereby increasing communication within the team [77]. For example, in a semi-structured interview, respondents mentioned that the flexibility of the leader encourages them to discuss and share actual ideas and problems with each other [79]. This enables the resolution of problems within the team and the project. The importance of communication in software project success is also discussed in Section 3.1.1.

Sharing information is an important indicator of project success and does not happen automatically [74]. Therefore, team managers should encourage teams to share information with one another and provide continuous learning opportunities. By offering training and learning opportunities, team managers enable the team to develop itself, thereby achieving technical excellence [48], which increases the chances of successful ASD practices [72]. Technical excellence is a fundamental principle in the Agile Manifesto and is highlighted in Principle 9, which states that technical excellence enhances agility [4].

The relationship between the team manager and stakeholders is essential [77]. This relationship ensures that the team understands stakeholders' needs and expectations, thereby enabling these needs to be met within the project. Additionally, this can help prevent potential problems or conflicts that might arise in the later stages of the project [77]. As a result, the project can be delivered successfully and with customer satisfaction in the end.

In conclusion, team managers have various aspects that can be crucial for achieving

project success. Different studies have examined the effect of team managers on project performance.

3.3 Customer

Since the publication of the Agile Manifesto, the customer's role in software development has gained increasing significance. The Agile Manifesto emphasizes collaborating with customers over negotiating contracts [4]. One of its principles states that the highest priority is to satisfy the customer through the early and continuous delivery of valuable software [4]. In this context, the involvement of customers throughout the agile software project life-cycle becomes inevitable.

3.3.1 Customer Involvement

Customer involvement refers to interactions between one or more representatives of the customer and the company throughout the project life- cycle [80].

Unlike traditional waterfall approaches, where customer involvement is often limited to the requirements phase, agile methodologies insist that the customers are present from the initial to the end phases of the project [81]. This ongoing collaboration helps promptly address changes and decreases the risk of project failure and overrun due to unmet customer expectations.

Throughout the software development process, minor and major changes often occur within the project. Customer involvement plays a vital role in clarifying the project's requirements, making it a critical factor for the success of software development projects. Additionally, customer involvement facilitates the early identification of project problems from the customer's perspective, enabling the team to address issues promptly and effectively, thereby reducing unnecessary costs.

An important point in customer involvement is the selection of the right people. Turner and Boehm mentioned that the customer should be represented by skilled

individuals who are collaborative, representative, authorized, committed, and knowledgeable (CRACK) [82]. They also indicate that a part-time CRACK performer is a better option than a full-time non-CRACK performer because deficiencies in any of these critical capabilities can lead to frustration, delays, wasted project resources, and an unacceptable final product [82].

Several studies have found a statistical relationship between project success and customer involvement [2, 3, 49]. For example, research by Cao and Chow identified customer involvement as one of the CSFs of a software project for the scope dimension, supporting its importance [2]. Tam et al. used partial least squares structural equation modeling (PLS-SEM) and found that their model could explain %46 of project success, identifying the critical factors as customer involvement (CI) and team capability (TC) [3]. Barros et al. also found a strong effect of customer involvement on the success of ASD [49]. Also, other closely related factors (e.g., customer collaboration, customer commitment, and competency) have also been identified as CSFs for agile software projects in former studies [2, 48].

3.4 Project

Projects are typically evaluated based on technical factors such as complexity, duration, and size, which significantly influence their success [5, 2, 51]. However, this thesis focuses on understanding how to follow a delivery strategy that aims to satisfy customers and maintain good relationships with them. The study explores the impact of a customer-oriented delivery strategy on software project success. This shift in focus aims to highlight the critical role of human factors in achieving project success different than traditional technical evaluations.

3.4.1 Delivery Strategy

Delivery strategy corresponds to multiple principles of the Agile Manifesto. Specially, the highest priority is to satisfy customers through early and continuous delivery of

valuable software and the frequent delivery of working software, with a preference for shorter timescales [4]. These principles emphasize the importance of maintaining a steady flow of deliverables that meet customer needs and expectations. However, the delivery strategy in traditional software development differs significantly from that of ASD. Traditional software development tends to have only one final delivery, well-documented and delivered several months after the initial discussions with the customer to define the requirements[49].

The classification of delivery strategy in the context of agile methodologies can vary. While some studies, such as Stankovic et al. and Chow and Cao, have evaluated it as a technical factor [2, 51]. On the other hand, this thesis interprets it as a human factor. This interpretation is based on the significant role of stakeholder feedback and interaction in the delivery process.

Chow and Cao's research indicates a statistically significant relationship between delivery strategy and the success attributes "Scope," "Timeliness," and "Cost" [2]. Among these, the delivery strategy has the most substantial effect on the success attributes examined in the research. Their findings suggest that delivery strategy can effectively manage project scope, ensure on-time completion, and stay within budget, leading to higher project success rates.

By classifying delivery strategy as a human factor, this thesis highlights the importance of ongoing communication and collaboration between the development team and stakeholders. This perspective aligns with agile principles, prioritizing individuals and interactions over processes and tools.

Considering that customer satisfaction is one of the critical success metrics for project performance, an early and continuous delivery strategy is expected to contribute significantly to project success. As mentioned earlier, agile project delivery advocates for frequent and early delivery as essential for achieving customer satisfaction within the project [4]. The agile-style delivery strategy ensures the project is

aligned with customer needs and expectations throughout development. Therefore, emphasis should be placed on the project's delivery strategy to increase the likelihood of success.

3.5 Software Methodologies

The definition of software methodologies is the structured approach used for planning, executing, and managing the process of software development. It has a set of principles, practices, procedures, and techniques that guide the development team in developing software. For example, principles, practices, and values of Extreme Programming can be seen in subsection 1.3.2. Software methodologies have an important role, and it is important to choose the proper methodology for the software project success. The findings of Joslin et al. reveal that there is a positive relationship between project methodology elements and the characteristics of project success [83]. Also, the other study mentioned that inappropriate methodology increases the risk of project failure by using data from 720 software projects [24]. Therefore it is added the research framework to evaluate its effect on project success. For this thesis, agile software methodology is preferred because it gives importance to human factors and is the most preferred methodology among survey respondents (see Figure 8).

3.5.1 Agile Software Methodology

Agile software methodology's principles, practices, and values were provided under Section 1.2 detail. According to Serrador et al., there is a correlation between the use of the agile approach and the reported project success: efficiency and stakeholder satisfaction [84].

In the research model, the questions in the group of software methodologies are used to detect that the respondent's team uses agile software methodology actually in their software project development and its effect on software project success.

CHAPTER IV

THE RESEARCH FRAMEWORK

This chapter provides designing of the research framework and survey.

4.1 Designing the Research Framework

The research framework was formed after an extensive systematic literature review, as shown in Figure 7. The research framework consists of five main parts: team, management, customer, project, and software methodologies. They are evaluated as human factors with their components. The statistical relationship between these five categories and project success is analyzed based on survey responses from participants in Turkey. This thesis research aims to provide and analyze the statistical relationship between human factors and software project success in Turkey. Thus, before starting a project, teams, and managers can determine what to prioritize or give importance to.

Project success is evaluated using time, budget, customer satisfaction, and self-satisfaction of the respondents chosen after an extensive systematic literature review. Time and cost are derived from the Iron Triangle Model. Quality was identified as the most controversial part of the Iron Triangle Model due to its subjectivity [35]. Customer satisfaction has gained prominence in project evaluation. Also, the research indicates that customer satisfaction is the top criterion for achieving project success, followed by the Iron Triangle components [45]. Therefore, customer satisfaction is included instead of quality. The respondents' self-satisfaction is also added for a broader evaluation of the project's success. The evaluation of software project success in this thesis consists of these four components.

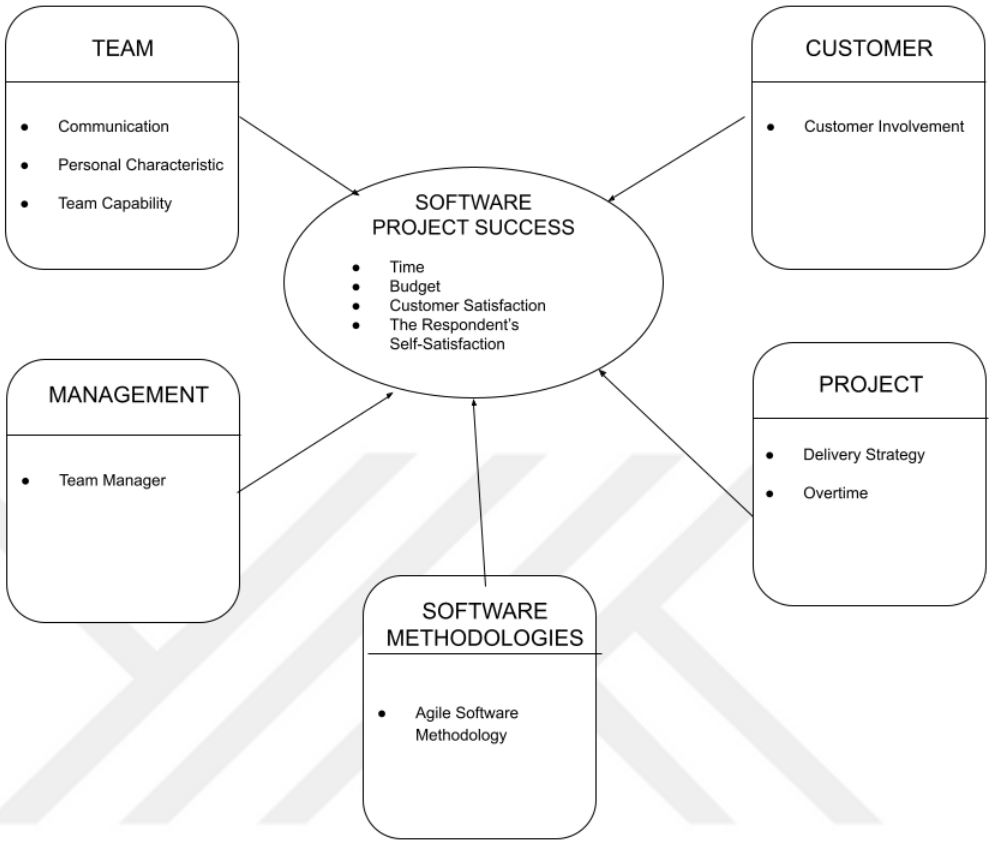

Figure 7: The research framework

4.2 Designing the Survey

Designing an effective survey is the most important step in collecting the necessary information to be used in research. The survey design process should be done carefully and should aim to collect the information needed to answer the research questions correctly. This includes defining clear objectives, selecting the appropriate type of survey, designing accurate and objective questions, and deciding the target population. In this study, the survey design process includes carefully considering these factors to obtain reliable and valid data to analyze human factors affecting software project success.

After conducting a comprehensive literature review, it was decided which human factors would be evaluated, focusing on software development teams, management,

customer involvement, software methodologies, and other project-related elements. Studies involving surveys were examined to inform this process, which was crucial in identifying and understanding the human factors affecting software project success. In particular, this study draws on insights from the work of Tam et al., as well as Misra et al. and Stankovic et al. [3, 48, 51]. The study was conducted in Turkey through a survey, with the next chapter providing a detailed discussion of the data. The survey aimed to evaluate the effects of these factors on the success of software projects with collected data. The original survey can be found in APPENDIX A.

CHAPTER V

METHODS

This chapter provides an overview of the data collection process and its analysis, including factor analysis, reliability and validity analysis, and Harman's Single-Factor test to check for common method bias in the data.

5.1 Data Collection

A survey was chosen for this thesis study to illustrate the effect of human factors on the success of software projects. The target population comprises individuals who are part of software development teams. Obtaining feedback from these individuals is crucial for the research.

The survey is in Turkish, and the respondents are members of software teams in Turkey. The survey comprises 63 questions, including multiple-choice, single- textbox, matrix/rating scale, and comment box formats. It is divided into seven sections: general questions, team, manager and management, software methodology, customer, project, and opinions. Survey questions are available in the Appendix A.

The survey questions use a 5-point Likert Scale for statistical calculations. Respondents rated their acceptance in percentages: %10, %30, %50, %70, and %90 to answer the questions accordingly. Responses are converted to a 1-5 scale for the analysis part.

The data for this thesis were collected through a survey conducted on the SurveyMonkey platform and shared across various channels, resulting in 312 respondents. Of these, 141 respondents completed the entire survey.

Table 9 provides a summary of the demographic characteristics of the survey respondents, focusing on their work experience and work roles within the software

Table 9: Demographic data

Work experience	**N**	**%**
0-5 years	28	%19.85
5-10 years	32	% 22.70
10-15 years	38	% 26.95
15-20 years	26	% 18.45
Over 20 years	17	% 12.05
Work role	**N**	**%**
Software Developer	55	% 39.00
Team Leader	27	% 19.15
Software Manager	11	%7.80
Analyst	11	% 7.80
Tester/Quality Controller	8	% 5.70
Project Manager	7	% 5.00
System Architect	6	% 4.25
Other	16	% 11.35

development teams. Figure 8 is a pie chart that used software methodologies by the survey respondents in their projects.

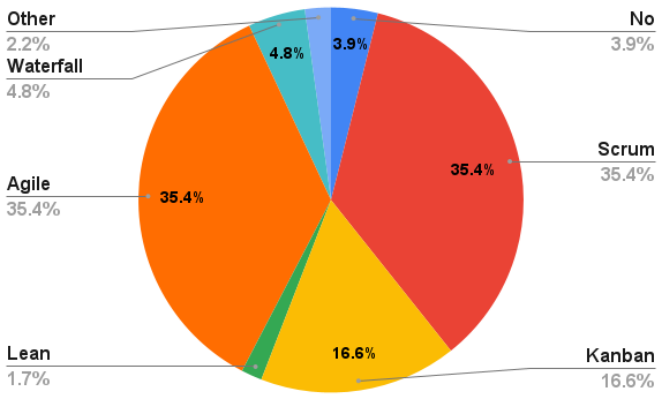

Figure 8: Preferred software methodologies of the survey respondents in their project

By analyzing the responses from individuals with different levels of experience and roles and using different software methodologies, we can gain insights into how human factors contribute to the success of software projects from different perspectives. This demographic data provides an essential understanding of the survey sample, ensuring that the findings are representative and comprehensive.

5.2 Data Analysis

This section presents a comprehensive analysis of the survey data collected to investigate the effect of human factors on the success of software projects. The analysis begins with factor analysis, followed by reliability and validity analysis. Finally, Harman's Single-Factor test was used as a common method of bias check.

5.2.1 Factor Analysis

A factor analysis was conducted to identify the underlying dimensions represented by the survey questions. The process involved several key steps.

- Dropping Dependent Variables

  The dependent variables should be removed before starting the factor analysis. Thus, only independent variables are included and used to identify the underlying factors.

- Deciding the Number of Factors:

  After a systematic literature review, the success of software projects decided to be examined under these five groups. Therefore, the survey questions were prepared and grouped based on team, project, management, software methodology, and customer. Factor analysis was used to reduce the number of questions in each group.

- Factor Analysis Technique:

Factor analysis was performed using the varimax rotation method, which was employed to simplify the factor structure by maximizing the variance of squared loadings of a factor across variables.

- Extracting Factor Loadings:

  The factor loadings were extracted from the analysis. Factor loadings are coefficients that describe the relationship between each observed variable and the underlying latent factors. They indicate how strongly each variable is associated with each factor.

- Interpreting Factor Loadings:

  The variables with the highest absolute value of loadings were identified for each factor. The top seven variables with the highest absolute loadings for team and project were examined to understand the common themes or concepts they represented. The top five variables with the highest absolute loadings are selected for the management, customer, and software methodologies. Based on these high-loading variables, each factor has updated versions in Turkish and English, which can be seen in Appendix B and Appendix C accordingly.

Table 10: Number of eliminated variables in each group

Construct	**Variable**
Team	6
Manager	2
Software Methodology	5
Customer	3
Project	0

By analyzing the factor loadings and identifying the variables most strongly associated with each factor, the underlying dimensions represented by the survey questions

were effectively uncovered. These insights contribute to a deeper understanding of the factors influencing project success in the context of the study.

5.2.2 Reliability & Validity Analysis

Reliability and validity analysis are important to ensure that the MLR and Ridge regression models are accurate and reliable. Reliability ensures consistency, while validity ensures that the model accurately predicts and measures the intended outcomes. Together, these analyses enhance the models' overall quality, credibility, and utility, making it a powerful tool for research and decision-making. Cronbach's alpha and composite reliability scores were calculated and evaluated for reliability analysis. The Fornell-Larcker criterion and common method bias (CMB) tests were preferred for validity.

Cronbach's alpha was introduced in the article "Coefficient Alpha and the Internal Structure of Tests" [85]. It is the most widely used estimator of test and scale reliability [86].

The Fornell-Larcker criterion compares the square root of the average variance extracted (AVE) for each construct with the correlations between that construct and all other constructs [87]. The criterion is satisfied if the square root of the AVE for a construct is greater than the correlation of that construct with any other construct in the model [87].

Table 11 presents the latent variables along with their mean, standard deviations, composite reliability (CR), Cronbach's alpha (CA), and validity values. All groups have CR values greater than 0.7, with many exceeding 0.8, indicating strong internal consistency [88]. Additionally, the diagonal values are the square root of the average variance extracted (AVE). They are represented in bold writing style. The satisfaction of the Fornell- Larcker criterion can be observed in Table 11 because the diagonal values are greater than the correlations among each pair of constructs (values in the

off-diagonal). Therefore, the proposed model demonstrates good internal consistency.

Specifically, the Team, Management, Software Methodology, and Customer constructs have Cronbach's alpha values greater than 0.7, confirming their reliability. Although the Success and Project constructs have Cronbach's alpha values less than 0.7 but greater than 0.6, they are still considered acceptable.

These evaluations demonstrate that the survey items are reliable and valid indicators of the intended constructs, enhancing the robustness and credibility of this study's findings.

Table 11: Latent variables mean, standard deviations (SD), composite reliability (CR), Cronbach's alpha (CA), and validity (AVE) measures.

Group	**Mean**	**SD**	**CR**	**CA**	**T**	**M**	**SM**	**C**	**P**	**S**
Team(T)	4.041	.937	.902	.868	**.719**					
Management(M)	3.829	1.193	.870	.912	.368	**.698**				
Software Meth.(SM)	3.349	1.476	.873	.743	.384	.472	**.684**			
Customer(C)	3.190	1.353	.876	.758	.253	.229	.305	**.672**		
Project(P)	3.912	1.231	.913	.616	.454	.529	.564	.382	**.731**	
Success(S)	3.819	1.033	.841	.671	.474	.410	.287	.228	.619	**.701**

5.2.3 Harman's Single-Factor Test for Common Method Bias

Common method bias (CMB) can pose a significant threat to the validity of survey research by distorting the observed relationships between variables. This bias arises when the measurement method introduces systematic error, thereby altering the true relationships among the constructs being measured. Sources of CMB include using the same method to assess different constructs, the wording and format of survey items, and the context in which the data is collected.

Given the context of software projects, controlling for CMB is particularly crucial, as human factors are often influenced by multiple overlapping constructs. To assess

the presence of CMB in the dataset, Harman's Single-Factor Test was conducted after performing factor analysis and reliability analysis to identify potential sources of bias.

The analysis revealed that the first factor extracted through factor analysis explained only %26.24 of the total variance, which is below the %50 threshold [89]. This finding suggests that CMB is not a major concern in the dataset [89], enhancing the validity of the survey results and supporting the robustness of subsequent analyses.

CHAPTER VI

RESULTS

This chapter presents two predictive models: Multiple Linear Regression (MLR) and Ridge regression applied to the survey data. The performance of both models is compared using R-squared (R^2), Root Mean Squared Error (RMSE), and Mean Square Error (MSE).

6.1 Multiple Linear Regression Model

MLR is a statistical method used to test and estimate the relationship between a dependent variable and one or more independent variables [90]. This research aims to explore the factors that can influence software project success, whether positively or negatively. These factors belong to customer, team, project, software methodology, and management, and each has subparts. Project success is evaluated as the average of time, budget, customer, and respondent's (own) satisfaction. They are represented as one dependent variable, which averages these four factors. Therefore, the MLR model is selected in this case.

According to McClave et al., the general form of the MLR model, which incorporates k independent variables can be expressed as [91]:

$$Y = \beta_0 + \beta_1 \cdot \mathrm{x1} + \beta_2 \cdot \mathrm{x2} + ... + \beta_k \cdot \mathrm{xk} + \epsilon$$

In this model, Y represents the dependent variable, while $X_1, X_2, \ldots, X_k$ are the independent variables. The coefficients β_i indicate the effect of each independent variable X_i when all other independent variables are held constant, with β_0 being the y-intercept. A positive coefficient indicates a positive relationship, while a negative

coefficient indicates a negative relationship. The term ϵ denotes the random error component.

In the case of this study, the above translates to the following general equation of the multiple regression model:

$$S(t, b, c, e) = 0.936 + 0.150 \cdot \mathrm{T} + 0.115 \cdot \mathrm{M} - 0.034 \cdot \mathrm{SM} + 0.009 \cdot \mathrm{C} + 0.490 \cdot \mathrm{P} + \epsilon \tag{2}$$

1. $S(t, b, c, e)$: Success with time, budget, and customer & employees (own) satisfaction
2. β_0: Intercept
3. β_i: Coefficients for each construct
4. T: Team M: Management SM: Software Methodology C: Customer P: Project
5. ϵ: Error term

Table 12: Multiple Linear Regression coefficients and p-values of the research model variables

Variable	**Coefficient**	p**-value**
Constant (Intercept)	0.936	0.009
Team	0.150	0.072
Management	0.1155	0.050
Methodology	-0.0341	0.548
Customer	0.0097	0.872
Project	0.4901	<0.001

In the regression analysis evaluating factors influencing software project success, the intercept is found to be significantly different from zero ($p = 0.009$), indicating a strong baseline success level when all predictors are zero. The reason is the intercept is calculated by giving zero to all predictor variables. The "Project" variable

demonstrates a highly significant effect on success ($p < 0.001$), underscoring the critical importance of project-specific attributes. Management practices with a marginal significance ($p = 0.050$) also contributes positively to project success, though the evidence is not overwhelmingly strong, indicating that effective management is beneficial but may need further support to be proven. The capabilities of the team show borderline significance ($p = 0.072$), suggesting that while team characteristics and skills might impact project success, additional data, or a slightly relaxed significance threshold could provide clearer confirmation of this effect.

On the other hand, the "Methodology" variable ($p = 0.548$) is not statistically significant, implying that the choice of software development methodology does not play a crucial role in project success within this model. In this model, the software methodology is agile software methodology, which is unexpected. This result indicates that there may be some missing parts. For the next research, can think of asking different questions to capture the agile software methodology. Similarly, the "Customer" variable ($p = 0.872$) is also not significant, indicating that variations in customer involvement do not significantly explain the variations in project success among the surveyed projects. It is also an unexpected result; many studies about the customer are statistically important for software project success [2, 3, 49]. Like software methodologies, there can be some possible problems or missing something when selecting and identifying customer group questions.

This analysis highlights the crucial importance of managing project-specific attributes and effective management practices while suggesting that team capabilities may also play a role. However, it indicates that the choice of development methodology and the level of customer involvement, as measured in this study, are not significantly critical to the success of software projects. These findings provide valuable insights for practitioners focusing on project and management practices to enhance the likelihood of successful project outcomes.

6.2 Ridge Regression Model

Ridge regression is a form of regularized regression that addresses multicollinearity by adding a penalty to the regression coefficient [92]. This penalty term prevents the coefficients from becoming too large, which enhances model generalizability, particularly in cases with many correlated independent variables. Ridge regression minimizes the sum of squared residuals and adds the square of the magnitude of the coefficients as a penalty term [92]. The primary objective is to balance model complexity and performance by controlling for overfitting, which often occurs in datasets with multicollinearity or small sample sizes. In this thesis research, the sample size can be considered as a small sample size so, the Ridge regression model is selected as a second model for this purpose.

Table 13: Ridge Regression coefficients of the research model variables

Variable	**Coefficient**
Team	0.1328
Management	0.0724
Methodology	-0.0128
Customer	-0.0130
Project	0.5757

The R-squared (R^2) score of the Ridge regression model is 0.287, indicating that it explains approximately %28.7 of the variance in software project success based on the independent variables. The model's RMSE is 0.514, and its MSE is 0.264. While Ridge regression does not typically provide p-values for the significance of individual coefficients, its ability to shrink coefficients makes it ideal for enhancing prediction robustness. Notably, the error score of the Ridge regression model is lower than that of the MLR model, suggesting that Ridge regression provides better accuracy. Coefficients of the variables in the Ridge regression model can be seen in Table 13.

Table 13 shows that the "Project" component of the research framework has the greatest impact on software project success, followed by the "Team" and "Management" components, similar to the MLR model. On the other hand, the "Customer" and "Management" components negatively affect project success, which is unusual. However, their coefficients are small and can be disregarded. These two components, particularly "Customer," have been shown to have a significant positive relationship with software project success in the literature [2, 3, 49]. There can be different reasons for this difference which is discussed under the Conclusion chapter.

6.3 Performance Comparison of the MLR and Ridge Regression Models

The R-squared (R^2) score of the MLR model is 0.358, indicating that it explains approximately %35.8 of the variance in software project success based on the independent variables. The R-squared (R^2) score of the Ridge regression model is 0.287, meaning it explains about %28.7 of the variance in software project success using the same independent variables.

The MSE of the MLR model is 0.287, representing the average of the squared differences between the observed and predicted values. Its RMSE is 0.535, measuring the average prediction error. This relatively low RMSE suggests reasonable model performance. However, RMSE and MSE can be difficult to interpret on their own; they are most useful when comparing different models. In this thesis, both MLR and Ridge regression models are used, and these scores are applied for comparison.

The RMSE of the Ridge regression model is 0.514, and its MSE is 0.264, both of which are lower than the scores for MLR. This indicates that Ridge regression is the better model for prediction accuracy, as it minimizes both prediction errors, making it more reliable for generalization.

The f-statistic score of MLR is 15.26. The F-statistic tests the overall significance of the model. It compares the model with no predictors (intercept only). A higher

F-statistic indicates that the model is better than a model with no predictors. F-statistic p-value is 6.61×10^{-12}. This very low p-value indicates that the overall model is statistically significant, meaning that the independent variables collectively impact the dependent variable (success). On the other hand, Ridge regression does not typically provide p-values or F-statistics due to regularization.

As a result, the MLR model explains a moderate portion of the variance in the success variable based on the R^2 score and has statistically significant predictors overall (F-statistic p-value 6.61×10^{-12}). The R^2 of MLR is higher than the Ridge regression model. This thesis aims to interpret relationships between human factors and software project success. Therefore, although the Ridge regression has a lower error score, the MLR is selected as the model because it explains more variance (higher R^2), indicating a greater ability to explain the variance in software project success.

However, the R^2 values for both models suggest other factors not accounted for that could further explain the variance in success. Additional human factors could be included in the models, or the sample size could be increased to improve the explanatory power.

CHAPTER VII

CONCLUSIONS

This thesis explores the use of two regression models, Multiple Linear Regression (MLR) and Ridge Regression, to assess the significance and impact of selected human factors on software project success within the research framework in the context of Turkey. While Ridge Regression demonstrated improved predictive accuracy, as indicated by lower RMSE and MSE values, the primary goal was to examine the relationship between human factors and project success. For this reason, MLR was chosen for its higher R^2 value, which better explains the variance in software success and allows for a clearer interpretation of individual factors. Although Ridge Regression improves predictive accuracy, this study focused on understanding the contributions of human factors to project outcomes. After evaluating both models, MLR was preferred for its superior ability to explain variance, as demonstrated by its higher R^2 score.

The results from the MLR model provide managers and teams with insights into which human factors should be considered significant. Consequently, they can allocate their resources and time accordingly, which enhances the likelihood of the project's success. According to the research model, the most important component is the "Project," followed by "Management" and "Team". Following customer-centric delivery strategies and avoiding or decreasing over time are important factors. Managers' relationship between both customer and software development is crucial. Also, managers should provide a suitable environment for software development. Personal characteristics, communication skills, and team capability should be considered during the formation of the software teams. Surprisingly, agile software methodology and

customer involvement are not statistically important for software project success in our model. The project questions included a component of the agile delivery strategy, which involves the customer in the delivery process. This may indicate a different aspect of customer involvement compared to the literature. According to the "Project" coefficient in the MLR model, customer involvement is more important specifically in the delivery process. Still, these two groups of questions can be reconsidered or found alternative questions.

An R^2 value in the range of 0.3 to 0.4 might be considered acceptable, as human behavior and outcomes are often influenced by numerous unpredictable factors. The research model explains %35.8 of the variance in software project success with the given independent variables, suggesting the model is moderately strong, particularly when considering the focus on human factors in this thesis.

Previous literature includes various research models that better explain software success with human factors, but these differences might stem from cultural or national variations or the nature of the respondents. Nevertheless, there is room for improvement in the model's performance. One approach could involve conducting interviews with managers and software development teams before designing the survey questions, allowing for a broader range of perspectives. Based on their feedback, new questions could be added, or existing ones refined. Another option is to pilot the survey with a small group of participants to gather feedback on the clarity and comprehensibility of the questions.

As a result, there is no "Silver Bullet" in software development. It is always open to changes and improvements. Because it is unrealistic to expect any process involving humans to remain constant or to have a single solution.

APPENDIX A

SURVEY QUESTIONS

Genel Sorular

Bu sayfada sizin hakkınızda sorulmuş soruları cevaplayınız.

1. Cinsiyetiniz nedir?

- ◯ Erkek
- ◯ Kadın
- ◯ Diğer (lütfen belirtin)

2. Doğum yılınızı girer misiniz?

3. Fiilen hangi şirkette veya organizasyonda çalışıyorsunuz?

4. Mesleğinizdeki kaçıncı yılınızdasınız?

5. Ekipteki rolünüz nedir?

- ◯ Yazılım Geliştiricisi
- ◯ Proje Yöneticisi
- ◯ Test/Kalite Kontrolcu
- ◯ Analist
- ◯ Sistem Mimarı
- ◯ Takım Lideri
- ◯ Yazılım Yöneticisi
- ◯ Ürün Yöneticisi
- ◯ Diğer (lütfen belirtin)

6. Son iki yılınızda kaç farklı ekip içinde rol aldınız?

Ekip

Bu sayfada ekibiniz ile ilgili soruları cevaplayınız.

1. Mevcut yazılım geliştirme ekibiniz kaç kişiden oluşuyor?

2. Ekip içindeki kişisel ilişkileri düşünürsek, bu ilişkilerin ne kadarı olumlu ve güçlü ilişkilerdir?

- ◯ %10
- ◯ %30
- ◯ %50
- ◯ %70
- ◯ %90

3. Ekibinizi değerlendirdiğinizde, çalıştığınız projelere göre ekibinizin teknik yeterliliğini hangi aralıkta görüyorsunuz?

%0-20	%20-40	%40-60	%60-80	%80-100
◯	◯	◯	◯	◯

4. Ekibinizdeki kişi sayısını göz önünde bulundurduğunuzda, toplantılarınıza düzenli olarak ekibinizin yaklaşık yüzde kaçı katılır?

%10	%30	%50	%70	%90
◯	◯	◯	◯	◯

5. Ekibinizdeki insanların, ne kadarının görev tanımları vardır ve nettir?

%10	%30	%50	%70	%90
◯	◯	◯	◯	◯

6. Projede karşılaştığım teknik problemleri ya da zorlukları aşağıdaki oranda ekip arkadaşlarımla paylaşıp tartışabilirim.

%10	%30	%50	%70	%90
◯	◯	◯	◯	◯

7. Ekibiniz, insan ilişkileri ve iletişimi iyi kişilerden oluşur.

	%10	%30	%50	%70	%90
Katılma oranınız:	◯	◯	◯	◯	◯

8. Ekibiniz, dürüst kişilerden oluşur.

	%10	%30	%50	%70	%90
Katılma oranınız:	◯	◯	◯	◯	◯

9. Ekibiniz, proje için istekli ve motivasyon sahibidir.

	%10	%30	%50	%70	%90
Katılma oranınız:	◯	◯	◯	◯	◯

10. Ekibiniz, işbirlikçi tutuma sahip olan kişilerden oluşur.

	%10	%30	%50	%70	%90
Katılma oranınız:	◯	◯	◯	◯	◯

11. Ekibiniz, sorumluluk sahibi kişilerden oluşur.

	%10	%30	%50	%70	%90
Katılma oranınız:	◯	◯	◯	◯	◯

12. Ekibiniz, öğrenmeye açık ve hazır insanlardan oluşur.

	%10	%30	%50	%70	%90
Katılma oranınız:	◯	◯	◯	◯	◯

13. Ekip çalışması, sizin performansınıza olumlu yönde katkı sağlar.

	%10	%30	%50	%70	%90
Katılma oranınız:	◯	◯	◯	◯	◯

14. Ekibiniz, uyumlu ve kendi kendini örgütleyen/düzenleyen(self-organizing) bir ekiptir.

	%10	%30	%50	%70	%90
Katılma oranınız:	◯	◯	◯	◯	◯

Yönetim ve Yönetici

Bu sayfada yöneticiniz ve ekibinizin yönetimi ile ilgisi soruları cevaplandırınız.

1. Ekip yöneticisi haricinde, ekibinizde proje yöneticiniz var mı?

◯ Evet

◯ Hayır

2. Projenin işleyişi ile ilgili görüşlerimi ya da gördüğüm problemleri, aşağıdaki oranda proje liderimle paylaşırım.

%10	%30	%50	%70	%90
◯	◯	◯	◯	◯

3. Ekip yöneticiniz, ekibinize esnek ve yaratıcı iş ortamı sağlar.

	%10	%30	%50	%70	%90
Katılma oranınız:	◯	◯	◯	◯	◯

4. Ekip yöneticiniz, ekibinizi sürekli öğrenme ve uygulama yolunda ilerletir.

	%10	%30	%50	%70	%90
Katılma oranınız:	◯	◯	◯	◯	◯

5. Ekip yöneticiniz, uyguladığınız metedoloji hakkında bilgi sahibidir ve başarılı şekilde de uygulamaktadır.

	%10	%30	%50	%70	%90
Katılma oranınız:	◯	◯	◯	◯	◯

6. Ekip yöneticiniz, proje boyunca desteğini ekibinize hissettirir.

	%10	%30	%50	%70	%90
Katılma oranınız:	◯	◯	◯	◯	◯

7. Ekip yöneticinizin, müşterilerinizle ilişkileri iyidir.

	%10	%30	%50	%70	%90
Katılma oranınız:	◯	◯	◯	◯	◯

8. Ekibinizin başarısında ya da başarısızlığında yöneticinizin payı ne kadardır?

%10	%30	%50	%70	%90
◯	◯	◯	◯	◯

Yazılım Metodolojisi

Bu sayfadaki soruları cevaplandırınız

1. Projelerinizde tercih ettiğiniz, belirli bir proje yönetimi metodolijisi var mı?

- ☐ Hayır
- ☐ Scrum
- ☐ Kanban
- ☐ Lean
- ☐ Agile
- ☐ Şelale(waterfall)
- ☐ Diğer (lütfen belirtiniz)

2. Bir önceki soruda belirtiğiniz metedolojilerin kurallarını ve prensiplerini, proje sürecinde yüzdelik olarak ne kadar uyguladığınızı düşünüyorsunuz?

%10	%30	%50	%70	%90
◯	◯	◯	◯	◯

3. Ekibinizde çevik süreçleri yürütmek ve iyileştirmekle görevli bir çalışan var mı (Scrum Master gibi)? Evet ise, bu göreve ne kadar zaman ayırıyor?

Yok	%25	%50	%75	Tam zamanlı
◯	◯	◯	◯	◯

4. Yazılım ekibi olarak günlük toplantı ...

Yapmıyoruz	15 dk sürüyor	yarım saat sürüyor	1 saat sürüyor	1 saati aşıyor
◯	◯	◯	◯	◯

5. Projeleri parçalara ayırıyorsanız (sprint gibi) yaklaşık ne kadar süre aralıklarına bölüyorsunuz?

Ayırmıyoruz	0-1 haftalık	1-2 haftalık	2-3 haftalık	3-4 haftalık	4 haftadan daha fazla
◯	◯	◯	◯	◯	◯

6. Projeyi bölerken, proje için harcayacağımız eforları önceden tahmin etmeye ya da belirlemeye çalışıyoruz.

	%10	%30	%50	%70	%90
Katılma oranınız:	◯	◯	◯	◯	◯

7. Proje esnasında, tamamlanmış kısımlar ile ilgili ekiple değerlendirme yapıyoruz.

	%10	%30	%50	%70	%90
Katılma oranınız:	◯	◯	◯	◯	◯

8. Proje esnasında, neleri nasıl yapabiliriz ya da geliştirebiliriz diye ekiple tartışıyoruz.

	%10	%30	%50	%70	%90
Katılma oranınız:	◯	◯	◯	◯	◯

9. Yazılım projesini, ekiple ortak bir alanda geliştiririz.

	%10	%30	%50	%70	%90
Katılma oranınız:	◯	◯	◯	◯	◯

10. Projede, zamanımızın aşağıdaki oranında birden fazla kişi aynı görev üzerinde birlikte çalışarak geçiririz.

%10	%30	%50	%70	%90
◯	◯	◯	◯	◯

11. Ekipte, birbirimizin kodlarına aşağıda seçtiğim cevap oranında hakimizdir.

%10	%30	%50	%70	%90
◯	◯	◯	◯	◯

Müşteri

Bu sayfadaki müşteri ve müşteri ilişkileri hakkındaki soruları cevaplayınız.

1. Çalıştığınız müşterileriniz, proje sürecine ne kadar dahil oluyor?

%10	%30	%50	%70	%90
◯	◯	◯	◯	◯

2. Müşterilerinizle iş ilişkinizi ve iletişiminizi düşünürsek, bunların ne kadarı olumlu ve güçlüdür?

%10	%30	%50	%70	%90
◯	◯	◯	◯	◯

3. Proje esnasında karşılaştığımız sorunlar, fark ettiğimiz eksiklikler ya da ek görüş ve önerilerimiz gibi detayları aşağıdaki oranda müşterimiz ile paylaşırız.

%10	%30	%50	%70	%90
◯	◯	◯	◯	◯

4. Müşteri ile iletişimimiz ... gerçekleşiyor.

Sadece proje başında ve sonunda	Günlük	Haftalık	Aylık	Bir ayı geçiyor
◯	◯	◯	◯	◯

5. Projenin tamamlanmış kısımlarını, müşteriye gösterip görüşünü alırız.

	%10	%30	%50	%70	%90
Katılma oranınız:	◯	◯	◯	◯	◯

6. Projede müşteriyi temsil etmede görevlendirilmiş kişi, karar verme konusunda tam yetkiye ve bilgiye sahiptir. Örnek olarak, projedeki değişiklikleri onaylama/onaylamama ya da önceliklendirmesini verebiliriz.

	%10	%30	%50	%70	%90
Katılma oranınız:	◯	◯	◯	◯	◯

7. Müşteri proje süreci boyunca vardır adeta ekibin üyesi haline gelir. Ekibimiz ile işbirliği halindedir.

	%10	%30	%50	%70	%90
Katılma oranınız:	◯	◯	◯	◯	◯

8. Projelerimizde önemli önceliklerimizden biri, müşterilerimizi memnun etmektir.

	%10	%30	%50	%70	%90
Katılma oranınız:	◯	◯	◯	◯	◯

Proje

Bu sayfada projeniz ve projenizin süreciyle ilgili sorulmuş soruları cevaplandırınız.

1. Projenin sonlarına doğru ek mesaiye kalış sıklığımız artıyor.

	%10	%30	%50	%70	%90
Katılma oranınız:	◯	◯	◯	◯	◯

2. Ek mesaiye kaldığınızda daha çok ne üzerinde çalışıyorsunuz?

- ◯ Projenin düzeltilmesi gereken kısımlarıyla
- ◯ Projenin tamamlanmamış kısımlarıyla
- ◯ Ek mesaiye kalmıyoruz

3. Son bir yıldaki çalışma günlerinizin, yaklaşık olarak yüzde kaçında fazla mesaiye kaldınız?

%10	%30	%50	%70	%90
◯	◯	◯	◯	◯

4. Projede, sorunlarla genellikle projenin ... karşılaşırız.

başlarında	ortalarında	sonlarında
◯	◯	◯

5. Projelerinizde, proje başarısına daha katkı sağladığını düşündüğünüz için üzerine daha fazla efor harcadığınız kısımlar mevcuttur.

	%10	%30	%50	%70	%90
Katılma oranınız:	◯	◯	◯	◯	◯

6. Proje süresince, ekip olarak toplam aşağıdaki, oranda fazla mesai yaptık.

%20 (günde 1.5 saat)	%40 (günde 3 saat)	%60 (günde 5 saat)	%80 (günde 5-6 saat)	%100 (günde 7-8 saat)
◯	◯	◯	◯	◯

7. Projelerinizin gereklilikleri ve istenilen özellikleri belirli ve nettir. Proje esnasında, değişiklikler olur fakat çok sık olan bir durum değildir.

	%10	%30	%50	%70	%90
Katılma oranınız:	◯	◯	◯	◯	◯

8. Projemizin tamamlanmış kısımlarını düzenli olarak teslim ederiz.

	%10	%30	%50	%70	%90
Katılma oranınız:	◯	◯	◯	◯	◯

9. Projenin önemli kısımlarını tamamlayıp teslim etmeye öncelik veririz.

	%10	%30	%50	%70	%90
Katılma oranınız:	◯	◯	◯	◯	◯

10. Proje süreçlerimizde iletişimimiz ve müzakarelerimiz çoğunlukla yüz yüze gerçekleşir.

	%10	%30	%50	%70	%90
Katılma oranınız:	◯	◯	◯	◯	◯

11. Proje, ilk planlandığı takvim süresine göre ne kadar zamanda tamamlandı? Planlanan süreye x dersek, projenin süreci t olursa, t için aşağıdaki aralıklardan hangisi daha uygunsa onu seçiniz.

t<=x	x<t<=1.2x	1.2x<t<=1.5x	1.5x<t<=2.5x	t>=2.5x :hiç bitmediyse de bu seçeneği işaretleyiniz.
◯	◯	◯	◯	◯

12. Proje hesaplanan ya da tahmin edilen bütçeye x dersek; proje sonunda, projeye harcanan bütçeye b dersek bunların ilişkisi nasıl olur?

b<=x	x<b<=1.2x	1.2x<b<=1.5x	1.5x<b<=2.5x	b>2.5x proje iptal edildiyse de bu seçeneği işaretleyiniz.
◯	◯	◯	◯	◯

13. Projenin başında belirlenen yükümlülükleri ve müşterinin isteklerini yerine getirir ve projeyi teslim ederiz.

	%10	%30	%50	%70	%90
Katılma oranınız:	◯	◯	◯	◯	◯

14. Bugüne kadar yer aldığınız tüm projeleri bir bütün olarak ele alsak, bunların toplam başarısı aşağıdakilerden hangisine en yakındır?

%10	%30	%50	%70	%90
◯	◯	◯	◯	◯

Ek Görüş

Ek görüş ve geri bildirimlerinizi paylaşmak isterseniz, bu soruları cevaplayınız.

1. Ek soru öneriniz ya da görüşünüz var ise bizimle paylaşır mısınız?

2. Bizim ile görüşlerinizi paylaşmak ve sorularımızı cevaplamak için görüşmek ister misiniz?

- ◯ Hayır
- ◯ Evet ise mail adresinizi bizimle paylaşır mısınız?

APPENDIX B

SURVEY QUESTIONS AFTER FACTOR ANALYSIS WITH THEIR GROUPS TURKISH

EKİP	İletişim	T1	Ekibiniz, insan ilişkileri ve iletişimi iyi kişilerden oluşur.
	Kişisel Özellik	T2	Ekibiniz, işbirlikçi tutuma sahip olan kişilerden oluşur.
		T3	Ekibiniz, sorumluluk sahibi kişilerden oluşur.
		T4	Ekibiniz, öğrenmeye açık ve hazır insanlardan oluşur.
		T5	Ekip içindeki kişisel ilişkileri düşünürsek, bu ilişkilerin ne kadarı olumlu ve güçlü ilişkilerdir?
		T6	Ekibiniz, uyumlu ve kendi kendini örgütleyen/düzenleyen(self-organizing) bir ekiptir.
	Takım Yeteneği	T7	Ekibinizi değerlendirdiğinizde, çalıştığınız projelere göre ekibinizin teknik yeterliliğini hangi aralıkta görüyorsunuz?
YÖNETİM	Yönetici	M1	Ekip yöneticiniz, ekibinize esnek ve yaratıcı iş ortamı sağlar.
		M2	Ekip yöneticiniz, ekibinizi sürekli öğrenme ve uygulama yolunda ilerletir.
		M3	Ekip yöneticiniz, uyguladığınız metedoloji hakkında bilgi sahibidir ve başarılı şekilde de uygulamaktadır.
		M4	Ekip yöneticiniz, proje boyunca desteğini ekibinize hissettirir.
		M5	Ekip yöneticinizin, müşterilerinizle ilişkileri iyidir.
YAZILIM METHODOLOJİLERİ	Çevik Yazılım Metodolojisi	SM1	Yazılım projesini, ekiple ortak bir alanda geliştiririz.
		SM2	Ekibinizde çevik süreçleri yürütmek ve iyileştirmekle görevli bir çalışan var mı (Scrum Master gibi)? Evet ise, bu göreve ne kadar zaman ayırıyor?
		SM3	Projede müşteriyi temsil etmede görevlendirilmiş kişi, karar verme konusunda tam yetkiye ve bilgiye sahiptir. Örnek olarak, projedeki değişiklikleri onaylama/onaylamama ya da önceliklendirmesini verebiliriz.
		SM4	Proje esnasında, tamamlanmış kısımlar ile ilgili ekiple değerlendirme yapıyoruz.
		SM5	Proje esnasında, neleri nasıl yapabiliriz ya da geliştirebiliriz diye ekiple tartışıyoruz.
MÜŞTERİ	Müşteri Katılımı	C1	Çalıştığınız müşterileriniz, proje sürecine ne kadar dahil oluyor?
		C2	Müşterilerinizle iş ilişkinizi ve iletişiminizi düşünürsek, bunların ne kadarı olumlu ve güçlüdür?
		C3	Proje esnasında karşılaştığımız sorunlar, fark ettiğimiz eksiklikler ya da ek görüş ve önerilerimiz gibi detayları aşağıdaki oranda müşterimiz ile paylaşırız.
		C4	Müşteri proje süreci boyunca vardır adeta ekibin üyesi haline gelir. Ekibimiz ile işbirliği halindedir.
		C5	Projenin tamamlanmış kısımlarını, müşteriye gösterip görüşünü alırız.
PROJE	Ek Mesai	P1	Projenin sonlarına doğru ek mesaiye kalış sıklığımız artıyor.
		P2	Son bir yıldaki çalışma günlerinizin, yaklaşık olarak yüzde kaçında fazla mesaiye kaldınız?
		P3	Proje süresince, ekip olarak toplam aşağıdaki, oranda fazla mesai yaptık.
	Teslim Stratejisi	P4	Projemizin tamamlanmış kısımlarını düzenli olarak teslim ederiz.
		P5	Projenin önemli kısımlarını tamamlayıp teslim etmeye öncelik veririz.
		P6	Projelerinizde, proje başarısına daha katkı sağladığını düşündüğünüz için üzerine daha fazla efor harcadığınız kısımlar mevcuttur.
		P7	Projenin tamamlanmış kısımlarını, müşteriye gösterip görüşünü alırız.
BAŞARI	Zaman	S1	Proje, ilk planlandığı takvim süresine göre ne kadar zamanda tamamlandı? Planlanan süreye x dersek, projenin süreci t olursa, t için aşağıdaki aralıklardan hangisi daha uygunsa onu seçiniz.
	Bütçe	S2	Proje hesaplanan ya da tahmin edilen bütçeye x dersek; proje sonunda, projeye harcanan bütçeye b dersek bunların ilişkisi nasıl olur?
	Müşteri Memnuniyeti	S3	Projenin başında belirlenen yükümlülükleri ve müşterinin isteklerini yerine getirir ve projeyi teslim ederiz.
	Katılımcının (kendi) memnuniyeti	S4	Bugüne kadar yer aldığınız tüm projeleri bir bütün olarak ele alsak, bunların toplam başarısı aşağıdakilerden hangisine en yakındır?

APPENDIX C

SURVEY QUESTIONS AFTER FACTOR ANALYSIS WITH THEIR GROUPS ENGLISH

TEAM	Communication	T1	Your team consists of people with strong interpersonal and communication skills.
	Personal Characteristics	T2	Your project team consisted of people who had a collaborative attitude.
		T3	Your project team consisted of people who had a sense of responsibility.
		T4	Your project team consisted of people who had the open and readiness to learn
		T5	Considering the personal relationships within your team, how many of these relationships are positive and strong?
		T6	Your team is harmonious and self-organizing.
	Team Capability	T7	When you evaluate your team, where do you see your team's technical competence in terms of the projects you work on?
MANAGEMENT	Manager	M1	Your team leader provides a flexible and creative work environment for your team.
		M2	Your team leader continuously guides your team towards learning and applying
		M3	Your team leader is knowledgeable about the methodology you use and applies it successfully.
		M4	Your team leader makes your team feel supported throughout the project.
		M5	Your team leader has good relations with your clients.
SOFTWARE METHODOLOGIES	Agile Software Methodology	SM1	Your team develop the software project in a shared space.
		SM2	Is there an employee in your team responsible for carrying out and improving agile processes (like a Scrum Master)? If yes, how much time do they devote to this role?
		SM3	The person assigned to represent the customer in the project has full authority and knowledge in decision-making. For example, they can approve or reject changes in the project or set priorities.
		SM4	During the project, you evaluate the completed parts with the team.
		SM5	During the project, you discuss with the team how and what you can do or improve.
CUSTOMER	Customer Involvement	CI1	How much do your customers get involved in the project process?
		CI2	Considering your work relationship and communication with your customers, how many of these are positive and strong?
		CI3	During the project, you share details such as the problems you encounter, deficiencies you notice, or additional opinions and suggestions with your customers to the following extent.
		CI5	The customer is present throughout the project and becomes almost a member of the team. They cooperate with your team.
		CI6	You show the completed parts of the project to the customer and get their feedback.
PROJECT	Overtime	P1	The frequency of working overtime increases towards the end of the project.
		P2	In the last year, approximately what percentage of your working days involved overtime?
		P3	Throughout the project, as a team, you did the following percentage of overtime.
	Delivery Strategy	P4	You regularly deliver completed parts of your project.
		P5	You prioritize completing and delivering the important parts of the project.
		P6	In your projects, there are parts on which you spend more effort because you think they contribute more to the project's success.
		P7	You show the completed parts of the project to the customer and get their feedback.
SUCCESS	Time	S1	How long did the project take compared to the initially planned schedule? If we say the planned duration is x, and the project duration is t, which of the following ranges best fits t?
	Budget	S2	If the calculated or estimated budget for the project is x, and the budget spent at the end of the project is b, what is the relationship between these two?
	Customer Satisfaction	S3	You fulfill customer requirements and requests determined at the beginning of the project and deliver the project.
	Respondent's (own) Satisfaction	S4	Considering all the projects you have been involved in so far as a whole, which of the following best describes their overall success?

REFERENCES

[1] J. F. Rockart, "A new approach to defining the chief executive's information needs," 1978.

[2] T. Chow and D.-B. Cao, "A survey study of critical success factors in agile software projects," *Journal of systems and software*, vol. 81, no. 6, pp. 961–971, 2008.

[3] C. Tam, E. J. da Costa Moura, T. Oliveira, and J. Varajão, "The factors influencing the success of on-going agile software development projects," *International Journal of Project Management*, vol. 38, no. 3, pp. 165–176, 2020.

[4] K. Beck, M. Beedle, A. Van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, *et al.*, "The agile manifesto," 2001.

[5] M. H. N. Nasir and S. Sahibuddin, "Critical success factors for software projects: A comparative study," *Scientific research and essays*, vol. 6, no. 10, pp. 2174–2186, 2011.

[6] A. Ahimbisibwe, R. Y. Cavana, and U. Daellenbach, "A contingency fit model of critical success factors for software development projects: A comparison of agile and traditional plan- based methodologies," *Journal of enterprise information management*, vol. 28, no. 1, pp. 7–33, 2015.

[7] M. Campbell-Kelly, "Development and structure of the international software industry, 1950-1990," *Business and Economic History*, vol. 24, 07 2008.

[8] B. Boehm, "A view of 20th and 21st century software engineering," in *Proceedings of the 28th international conference on Software engineering*, pp. 12–29, 2006.

[9] C. Baum, *The system builders: The story of SDC*. System Development Corporation, 1981.

[10] A. Kakar and A. Kakar, "A brief history of software development and manufacturing," in *SAIS 2020 Proceedings*, 2020.

[11] W. Aspray, R. Keil-Slawik, and D. Parnas, "The history of software engineering," tech. rep., Citeseer, 1996.

[12] NASA Science, "Margaret hamilton: Biography and contributions," 2024. Accessed: 2024-10-09.

[13] Y. Bassil, "A simulation model for the waterfall software development life cycle," *arXiv preprint arXiv:1205.6904*, 2012.

[14] B. W. Boehm, *Software and its impact: A quantitative assessment.* Rand Corporation, 1972.

[15] K. Crowston, K. Wei, J. Howison, and A. Wiggins, "Free/libre open-source software development: What we know and what we do not know," *ACM Computing Surveys (CSUR)*, vol. 44, no. 2, pp. 1–35, 2008.

[16] M. Poppendieck and M. A. Cusumano, "Lean software development: A tutorial," *IEEE software*, vol. 29, no. 5, pp. 26–32, 2012.

[17] T. Clancy, "The standish group report," *Chaos report*, 1995.

[18] R. N. Charette, "Why software fails [software failure]," *IEEE spectrum*, vol. 42, no. 9, pp. 42–49, 2005.

[19] Cambridge Business English Dictionary, "Agile." Online, 2011. Definition of "agile".

[20] K. Schwaber and M. Beedle, *Agile software development with Scrum.* Prentice Hall PTR, 2001.

[21] K. Schwaber, *Agile project management with Scrum.* Microsoft press, 2004.

[22] K. Beck, *Extreme programming explained: embrace change.* addison-wesley professional, 2000.

[23] M. O. Ahmad, J. Markkula, and M. Oivo, "Kanban in software development: A systematic literature review," in *2013 39th Euromicro conference on software engineering and advanced applications*, pp. 9–16, IEEE, 2013.

[24] A. Tiwana and M. Keil, "The one-minute risk assessment tool," *Communications of the ACM*, vol. 47, no. 11, pp. 73–77, 2004.

[25] K. Ewusi-Mensah, *Software development failures.* Mit Press, 2003.

[26] The Standish Group International, "Chaos report," 1994–2009.

[27] L. Alves, G. Souza, P. Ribeiro, and R.-J. Machado, "Longevity of risks in software development projects: a comparative analysis with an academic environment," *Procedia Computer Science*, vol. 181, pp. 827–834, 02 2021.

[28] R. Glass, "Software runaways-lessons learned from massive software project failures. 1998."

[29] A. J. Swartz, "Airport 95: Automated baggage system?," *ACM SIGSOFT Software Engineering Notes*, vol. 21, no. 2, pp. 79–83, 1996.

[30] M. Dowson, "The ariane 5 software failure," *ACM SIGSOFT Software Engineering Notes*, vol. 22, no. 2, p. 84, 1997.

[31] J.-L. Lions, L. Luebeck, J.-L. Fauquembergue, G. Kahn, W. Kubbat, S. Levedag, L. Mazzini, D. Merle, and C. O'Halloran, "Ariane 5 flight 501 failure report by the inquiry board," 1996.

[32] F. Brooks and H. Kugler, *No silver bullet*. April, 1987.

[33] Cambridge English Dictionary, "Success." Online, 2011. Definition of "success".

[34] R. Atkinson, "Project management: cost, time and quality, two best guesses and a phenomenon, its time to accept other success criteria," *International journal of project management*, vol. 17, no. 6, pp. 337–342, 1999.

[35] J. Pollack, J. Helm, and D. Adler, "What is the iron triangle, and how has it changed?," *International journal of managing projects in business*, vol. 11, no. 2, pp. 527–547, 2018.

[36] J. K. Pinto, *Project management: achieving competitive advantage*. Pearson, 2020.

[37] R. Turner and M. Huemann, "Managing quality," *Gower Handbook of Project management. Gower Publishing*, 2002.

[38] C. J. Van Wyngaard, J.-H. C. Pretorius, and L. Pretorius, "Theory of the triple constraint—a conceptual review," in *2012 IEEE International Conference on Industrial Engineering and Engineering Management*, pp. 1991–1997, IEEE, 2012.

[39] B. Van Der Hoorn and S. J. Whitty, "Signs to dogma: A heideggerian view of how artefacts distort the project world," *International Journal of Project Management*, vol. 33, no. 6, pp. 1206–1219, 2015.

[40] R. Turner and R. Zolin, "Forecasting success on large projects: developing reliable scales to predict multiple perspectives by multiple stakeholders over multiple time frames," *Project management journal*, vol. 43, no. 5, pp. 87–99, 2012.

[41] A. de Wit, "Measurement of project success," *International Journal of Project Management*, vol. 6, no. 3, pp. 164–170, 1988.

[42] M. Bronte-Stewart, "Beyond the iron triangle: Evaluating aspects of success and failure using a project status model," *Computing & Information Systems*, vol. 19, no. 2, pp. 19–36, 2015.

[43] C. Zid, N. Kasim, and A. R. Soomro, "Effective project management approach to attain project success, based on cost-time-quality," *International Journal of Project Organisation and Management*, vol. 12, no. 2, pp. 149–163, 2020.

[44] K. Davis, "Different stakeholder groups and their perceptions of project success," *International journal of project management*, vol. 32, no. 2, pp. 189–201, 2014.

[45] A. J. Shenhar, O. Levy, and D. Dvir, "Mapping the dimensions of project success," *Project management journal*, vol. 28, pp. 5–13, 1997.

[46] J. F. Rockart, "Chief executives define their own data needs.," *Harvard business review*, vol. 57, no. 2, pp. 81–93, 1979.

[47] J. K. Pinto and P. Rouhiainen, *Building customer-based project organizations*. John Wiley & Sons, 2002.

[48] S. C. Misra, V. Kumar, and U. Kumar, "Identifying some important success factors in adopting agile software development practices," *Journal of systems and software*, vol. 82, no. 11, pp. 1869–1890, 2009.

[49] L. Barros, C. Tam, and J. Varajão, "Agile software development projects–unveiling the human-related critical success factors," *Information and Software Technology*, vol. 170, p. 107432, 2024.

[50] T. DeMarco, "Looking for lost keys," *Software Magazine*, vol. 8, no. 5, pp. 58–62, 1988.

[51] D. Stankovic, V. Nikolic, M. Djordjevic, and D.-B. Cao, "A survey study of critical success factors in agile software projects in former yugoslavia it companies," *Journal of Systems and Software*, vol. 86, no. 6, pp. 1663–1678, 2013.

[52] V. Lalsing, S. Kishnah, and S. Pudaruth, "People factors in agile software development and project management," *International Journal of Software Engineering & Applications*, vol. 3, no. 1, p. 117, 2012.

[53] P. McBreen, *Software craftsmanship: The new imperative*. Addison-Wesley Professional, 2002.

[54] E.-M. Kweku, *Software Development Failures*. The MIT Press, 2003.

[55] R. E. Kraut and L. A. Streeter, "Coordination in software development," *Communications of the ACM*, vol. 38, no. 3, pp. 69–82, 1995.

[56] A. Bustamante and R. Sawhney, "Agile xxl: Scaling agile for project teams, seapine software," 2011.

[57] P. Abilla, "Team dynamics: Size matters redux," 2006.

[58] A. Salman, M. Jaafar, S. Malik, D. Mohammad, and S. A. Muhammad, "An empirical investigation of the impact of the communication and employee motivation on the project success using agile framework and its effect on the software development business," *Business Perspectives and Research*, vol. 9, no. 1, pp. 46–61, 2021.

[59] M. Pikkarainen, J. Haikara, O. Salo, P. Abrahamsson, and J. Still, "The impact of agile practices on communication in software development," *Empirical Software Engineering*, vol. 13, pp. 303–337, 2008.

[60] A. Begel and N. Nagappan, "Pair programming: what's in it for me?," in *Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement*, pp. 120–128, 2008.

[61] J. A. Espinosa and E. Carmel, "The impact of time separation on coordination in global software teams: a conceptual foundation," *Software Process: Improvement and Practice*, vol. 8, no. 4, pp. 249–266, 2003.

[62] S. Sawyer, J. Farber, and R. Spillers, "Supporting the social processes of software development," *Information Technology & People*, vol. 10, no. 1, pp. 46–62, 1997.

[63] S. Teasley, L. Covi, M. S. Krishnan, and J. S. Olson, "How does radical collocation help a team succeed?," in *Proceedings of the 2000 ACM conference on Computer supported cooperative work*, pp. 339–346, 2000.

[64] D. G. Ancona and D. F. Caldwell, "Demography and design: Predictors of new product team performance," *Organization science*, vol. 3, no. 3, pp. 321–341, 1992.

[65] D. Rodríguez, M. Sicilia, E. García, and R. Harrison, "Empirical findings on team size and productivity in software development," *Journal of Systems and Software*, vol. 85, no. 3, pp. 562–570, 2012.

[66] F. P. Brooks, *The Mythical Man-Month:Essays on Software Engineering*. New York: Addison-Wesley, 1975.

[67] T. W. Malone and K. Crowston, "The interdisciplinary study of coordination," *ACM Computing Surveys (CSUR)*, vol. 26, no. 1, pp. 87–119, 1994.

[68] P. S. DeOrtentiis, J. K. Summers, A. P. Ammeter, C. Douglas, and G. R. Ferris, "Cohesion and satisfaction as mediators of the team trust–team effectiveness relationship: An interdependence theory perspective," *Career Development International*, vol. 18, no. 5, pp. 521–543, 2013.

[69] A. E. Akgün, H. Keskin, J. Byrne, and S. Z. Imamoglu, "Antecedents and consequences of team potency in software development projects," *Information & Management*, vol. 44, no. 7, pp. 646–656, 2007.

[70] H. Imam and M. K. Zaheer, "Shared leadership and project success: The roles of knowledge sharing, cohesion and trust in the team," *International journal of project management*, vol. 39, no. 5, pp. 463–473, 2021.

[71] P. Yetton, A. Martin, R. Sharma, and K. Johnston, "A model of information systems development project performance," *Information Systems Journal*, vol. 10, no. 4, pp. 263–289, 2000.

[72] M. Lindvall, V. Basili, B. Boehm, P. Costa, K. Dangle, F. Shull, R. Tesoriero, L. Williams, and M. Zelkowitz, "Empirical findings in agile methods," in *Extreme Programming and Agile Methods—XP/Agile Universe 2002: Second XP*

Universe and First Agile Universe Conference Chicago, IL, USA, August 4- -7, 2002 Proceedings 2, pp. 197–207, Springer, 2002.

[73] M. R. Haas, "Knowledge gathering, team capabilities, and project performance in challenging work environments," *Management Science*, vol. 52, no. 8, pp. 1170–1184, 2006.

[74] G. Purna Sudhakar, A. Farooq, and S. Patnaik, "Soft factors affecting the performance of software development teams," *Team Performance Management: An International Journal*, vol. 17, no. 3/4, pp. 187–205, 2011.

[75] Project Management Institute, *A Guide to the Project Management Body of Knowledge (PMBOK® Guide)*. Newtown Square, PA: Project Management Institute, 7th ed., 2021.

[76] C. Jones, "Software project management practices: Failure versus success," *CrossTalk: The Journal of Defense Software Engineering*, vol. 17, no. 10, pp. 5–9, 2004.

[77] V. S. Anantatmula, "Project manager leadership role in improving project performance," *Engineering management journal*, vol. 22, no. 1, pp. 13–22, 2010.

[78] T.-P. Liang, C.-C. Liu, T.-M. Lin, and B. Lin, "Effect of team diversity on software project performance," *Industrial Management & Data Systems*, vol. 107, no. 5, pp. 636–653, 2007.

[79] A. Alami, O. Krancher, and M. Paasivaara, "The journey to technical excellence in agile software development," *Information and Software Technology*, vol. 150, p. 106959, 2022.

[80] P. Carbonell, A. I. Rodríguez-Escudero, and D. Pujari, "Customer involvement in new service development: An examination of antecedents and outcomes," *Journal of product innovation management*, vol. 26, no. 5, pp. 536–550, 2009.

[81] J. A. Highsmith, *Agile software development ecosystems*, vol. 13. Addison-Wesley Professional, 2002.

[82] B. Boehm and R. Turner, "Observations on balancing discipline and agility," in *Proceedings of the Agile Development Conference, 2003. ADC 2003*, pp. 32–39, IEEE, 2003.

[83] R. Joslin and R. Müller, "The impact of project methodologies on project success in different project environments," *International journal of managing projects in business*, vol. 9, no. 2, pp. 364–388, 2016.

[84] P. Serrador and J. K. Pinto, "Does agile work?—a quantitative analysis of agile project success," *International journal of project management*, vol. 33, no. 5, pp. 1040–1051, 2015.

[85] L. J. Cronbach, "Coefficient alpha and the internal structure of tests," *psychometrika*, vol. 16, no. 3, pp. 297–334, 1951.

[86] R. A. Peterson and Y. Kim, "On the relationship between coefficient alpha and composite reliability.," *Journal of applied psychology*, vol. 98, no. 1, p. 194, 2013.

[87] C. Fornell and D. F. Larcker, "Evaluating structural equation models with unobservable variables and measurement error," *Journal of marketing research*, vol. 18, no. 1, pp. 39–50, 1981.

[88] D. W. Straub, "Validating instruments in mis research," *MIS quarterly*, pp. 147–169, 1989.

[89] H. H. Harman, *Modern factor analysis*. University of Chicago press, 1976.

[90] G. K. Uyanık and N. Güler, "A study on multiple linear regression analysis," *Procedia-Social and Behavioral Sciences*, vol. 106, pp. 234–240, 2013.

[91] J. T. McClave, P. G. Benson, and T. Sincich, *Statistics for business and economics*. Pearson Education, 2008.

[92] G. C. McDonald, "Ridge regression," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 1, no. 1, pp. 93–100, 2009.

VITA

Nurver Şeyma Sel received her Bachelor of Science in Computer Science with a minor in Psychology from Sabancı University in 2021. After that, she continued her academic journey as a Master's student at Özyeğin University. She has been working as a Teaching Assistant at the Department of Computer Science under the supervision of Dr. Yaşar Safkan since September 2021. Her research studies focus on software project management and agile software development.