A NOVEL SOFTWARE DEVELOPMENT METHODOLOGY FOR RESEARCH-BASED SOFTWARE PROJECTS

A THESIS SUBMITTED TO

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

OF

ATILIM UNIVERSITY

BY

İBRAHİM CERECİ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR

THE DEGREE OF DOCTOR OF PHILOSOPHY

IN

THE DEPARTMENT OF SOFTWARE ENGINEERING

FEBRUARY 2019

Approval of the Graduate School of Natural and Applied Sciences, Atilim University.

_____

Prof. Dr. Ali Kara

Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of **Doctor of Philosophy in Software Engineering, Atilim University**.

_____

Prof. Dr. Ali Yazıcı

Head of Department

This is to certify that we have read the thesis "A Novel Software Development Methodology For Research-Based Software Projects" submitted by "İbrahim Cereci" and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Doctor of Philosophy.

_____

Asst. Prof. Dr. Ziya Karakaya

Supervisor

Examining Committee Members:

Prof. Dr. Ali Yazıcı _____
Software Engineering, Atılım University

Prof. Dr. Erdoğan Doğdu _____
Computer Engineering, Çankaya University

Asst. Prof. Dr. Erol Özçelik _____
Psychology Department, Çankaya University

Asst. Prof. Dr. Serhat Peker _____
Software Engineering, Atılım University

Asst. Prof. Dr. Ziya Karakaya _____
Computer Engineering, Atılım University

Date: 01.02.2019

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name:

Signature:

ABSTRACT

A NOVEL SOFTWARE DEVELOPMENT METHODOLOGY FOR RESEARCH-BASED SOFTWARE PROJECTS

Cereci, İbrahim

Ph.D. Candidate, Software Engineering Department

Supervisor: Asst. Prof. Dr. Ziya Karakaya

February 2019, 124 pages

Software development in mid-sized or large-scale projects is usually carried by a group of individuals, whose coordination, choosing suitable development practices for the group and keeping track of the software development process are all challenging tasks. Software development methodologies are heavily utilized for these purposes. Although previously proposed methodologies manage to meet the needs of the industry, they are not tailored to do so for academicians developing research-based software projects at universities. Therefore, in this thesis, our first aim is to show the necessity of a new software development methodology for research-based projects carried by the universities. Then, through current literature, interviews and questionnaires, the needs and the best practices of research-based projects are collected using grounded theory qualitative method. Finally, after completing the analysis of the findings, a new software development methodology is proposed tailored to the needs of the researchers that are working on research-based software projects. The proposed methodology is evaluated by the experts and it is found useful for the research-based software projects. Collected issues and best-practices are proposed as a guideline for the project managers, team members, funding agencies, universities, and the end-users of research-based software projects. These guidelines can be utilized to increase the productiveness of such projects.

Keywords: Software Development Methodology, Research-Based Software Project, Software Engineering, Project Management

ÖZ

ARAŞTIRMA TABANLI YAZILIM PROJELERİ İÇİN YENİ BİR YAZILIM GELİŞTİRME METODOLOJİSİ

Cereci, İbrahim

Doktora Derecesi Adayı, Yazılım Mühendisliği Bölümü

Tez Yöneticisi: Dr. Öğretim Üyesi Ziya Karakaya

Şubat 2019, 124 sayfa

Orta ve büyük ölçekli yazılım geliştirme projeleri genellikle bir çok takım üyesi tarafından birlikte geliştirilirler. Takım üyelerinin koordine edilmesi, grup için uygun geliştirme yöntemleri kullanılması ve grubun yazılım geliştirme sürecinin kontrolü zor problemlerdir. Yazılım geliştirme metotları yoğunlukla bu zor problemlerin çözümü için kullanılmaktadırlar. Var olan yazılım geliştirme metotları her ne kadar endüstrinin ihtiyaçlarını karşılasalar dahi, akademisyenler tarafından üniversitelerde yürütülen araştırma tabanlı yazılım geliştirme proje ihtiyaçlarını sağlamaya yönelik değillerdir. Bu çalışmada; araştırma tabanlı yazılım projeleri için yeni bir metotun gerekli olduğunu ortaya koymak adına, bu alanda çalışan kişilerin ihtiyaçları ve üstün yöntemleri nitel bir çalışma ile toplanıp, toplanan veriler ışığında da, araştırma tabanlı yazılım projeleri için yeni bir yazılım geliştirme metotu sunulmuştur. Önerilen metot alan uzmanlarının sağladığı uzman görüşleri ile değerlendirilip araştırma tabanlı yazılım projeleri için uygun bulunmuştur. Çalışma sırasında toplanmış olan alan problemleri ve üstün yöntemler, araştırma tabanlı yazılım projelerinde yer almak isteyen proje yöneticileri, takım üyeleri, üniversiteler ve destekleyici kurumlara öneriler olarak bir çerçevede sunulmuştur. Bu öneriler kullanılarak ilerde gerçekleştirilecek olan benzeri projelerin verimlilikleri artırılabilir.

Anahtar Kelimeler: Yazılım Geliştirme Süreci, Araştırma Tabanlı Yazılım Projesi, Yazılım Mühendisliği, Proje Yönetimi

To My Parents

ACKNOWLEDGMENTS

I would like to express sincere appreciation to my supervisor Asst. Prof. Dr. Ziya Karakaya for his guidance and support through the thesis period. Thanks also go to my family who supported me the entire time.

TABLE OF CONTENTS

LIST OF TABLES

TABLE

LIST OF FIGURES

LIST OF ABBREVIATIONS

AR	-	Action Research
COTS	-	Commercial Off-The-Shelf
CSR	-	Case Study Research
EBM	-	Evidence-Based Medicine
EBSE	-	Evidence-Based Software Engineering
GSD	-	Global Software Development
GT	-	Grounded Theory
OSS	-	Open Source Software
PM	-	Project Manager
RAD	-	Rapid Application Development
RBAgile	-	Research-Based Agile Software Development Methodology
RBSP	-	Research-Based Software Project
R&D	-	Research and Development
SDM	-	Software Development Methodology
SE	-	Software Engineering
XP	-	Extreme Programming

CHAPTER 1

INTRODUCTION

1.1 Motivation

Software Engineering (SE) is a complex field, and such complexity comes from both technological and human-related aspects. Human behavior, interaction, and communication are at the center of SE studies [1]. Usually, most software projects consist of numerous members working in parallel. To monitor the process and to ease the development of the software, developers utilize Software Development Methodologies (SDM) designed to help decide about the project lifecycle and how the development phase will progress. There is a variety of common software development methodologies proposed to fulfill the specific needs of the industry as per the conditions. In this way, each SDM has its strengths and weaknesses, and no single methodology can cover the needs of all the possible software projects.

Although there are many software development methodologies present, none of them are specifically tailored to the needs of research-based software projects carried out in the university environment [2] - an apparent gap in the literature. Industry projects and funded university projects have some fundamental differences. Focus on research in the universities versus focus on the product in the industry projects is one of the most apparent differences, but not the only one. Apart from this, the commercialization of university research has brought about specific obstacles not present in the industry [3]. The literature is not extensive concerning the differences between these two, or on the shortcomings of university projects, save a handful. See [2] [4] [5].

Research-Based Software Projects (RBSPs) are those that involve substantial research and development (R&D) at university environments and are managed by academicians. In this study, RBSPs are sometimes mentioned as university projects, academic projects or projects carried by academicians. RBSPs do not merely consist of software projects and include those from other disciplines with software incorporated therein. The literature on research-based software development is rare.

To the best of our knowledge, only one research group has focused on the importance of this problem extensively. In the present work, we aim to propose a software development methodology that is designed explicitly for RBSPs that are carried at universities.

1.2 Research Questions

The main research problem is directly related to research-based software projects done in universities. Currently, there is no specific software development methodology available to researchers to develop their RBSPs. Our strategy is to identify the problems faced by team members working on such projects, to find the best practices that apply to these types of projects, and to propose an SDM that is easily and effectively usable by researchers working on RBSPs. To achieve that, the following research questions are formed:

1. At what level, the common software development methodologies are effectively implemented in research-based software projects?

The authors have determined from the literature that, in the early phases, projects developed by universities generally do not employ common development methodologies effectively. This motivated us to identify the specific needs of researchers and how they approach the development process to come up with a suitable SDM for their purposes.

2. What are the common problems faced by the project team while working on research-based software projects?

There are limited studies that show the problems faced during the development of RBSPs. The authors' aim with this research question is to identify the common challenges and issues by using two different fact-finding techniques, namely interviewing and surveying, applied to individuals that have participated in such projects.

3. What are the best practices that are employed in research-based software projects?

Gathering the best practices from the academicians who have participated in research-based projects is essential for anyone who aims to develop a methodology that is both practical and suitable for the needs of individuals working on RBSPs.

Based on the above questions, the purpose of this study is to develop and propose a new software development methodology that will satisfy the needs of individuals involved in research-based software projects at universities.

1.3 The significance of the Study

Both the author and the thesis supervisor have been previously involved in software development projects in their universities and noticed first-hand that when SDMs are not utilized, those projects tend not to reach their potential. To increase the efficiency of RBSPs and not to waste the resources supplied by the national and international funding agencies, suitable SDMs need to be employed. In the present work developed as the first of its kind to the best of our knowledge, it is expected that the methodology to increase the productivity and improve the software development process in the long run for universities. As a result, these institutions, researchers and the funding agencies are likely to gain more efficiency. Additionally, stating the issues that are faced by the project team during the development of RBSPs, and providing guidelines for all the interested parties will allow these universities, funding agencies, and other researchers to understand the problemmatic areas better and propose new solutions accordingly.

1.4 Methodology

According to [6] and [7], statistics and quantitative methods cannot describe phenomena adequately if the subject area contains the complexity of human behavior. In page 11 of [7], there is a quote from LaPiere [8] that states “The study of human behavior is time-consuming, intellectually fatiguing, and depends for its success upon the ability of the investigator. Quantitative measurements are quantitatively accurate; qualitative evaluations are always subject to the errors of human judgment. It would seem far more worthwhile to make a shrewd guess regarding that which is essential than to accurately measure that which is likely to prove irrelevant.” This is also true in the present study. Since software development

projects include human behavior, interaction, and relations in their core, qualitative research methodologies are accepted as more suitable for SE research.

Here, qualitative research has been carried by the authors for understanding and categorizing the problems faced by team members and their best practices in RBSPs. Different qualitative research techniques have been considered, and consequently, Grounded Theory (GT) [9] is found to be the most suitable alternative. GT is an inductive method that utilizes the data to the extent of forming theories [10]. Data for GT is usually collected through interviews, field notes, and memos [11]. After data collection, the next step is to transcribe the data that is collected via voice or video recording. Then, through open coding, transcribed data is scanned, and the concepts are categorized, or in other words "coded." In later steps of the research, the constructed categories are refined, and their relations are driven. Finally, one or more stories can be driven from the data and personal memos, to develop theories that are grounded in the actual data which was collected at the beginning.

To gather the needs and best practices of the individuals that have participated in RBSPs, interviews are arranged with twenty interviewees and, later, the findings of these interviews are validated with a questionnaire that is applied to forty-seven academicians during the 3rd International Conference on Computer Science and Engineering (UBMK) held in 2018 in Sarajevo/Bosnia and Herzegovina. Based on the literature and the result of our research, a new software development methodology is proposed that is iteratively improved and validated with the help of expert opinions. The proposed method appears in Chapter 5, and the details about the research steps, research methods, and the data sets can be found in Chapter 3, Research Design & Methodology.

1.5 Limitations and Boundaries

This study is only related to the software projects that are research-based and carried out at universities and by academicians. Any other projects that involve the industry are out of the scope of this study. The issues, challenges, and the best practices of the team members are collected through interviews and questionnaires. Data collection is limited to these two sources. Convenience sampling is applied in this study so that it can be considered as a bias. The selected sample size of the study (20) is large

enough to generalize to universities not included in this study. The interviews are mainly conducted with academicians working in Turkey. Four of the academicians were working on international RBSPs. All the forty-seven participants of the questionnaire were academicians who were attendees of an international conference in the field of computer science.

The study aims to focus on certain parts of the software development and project management phases and to propose a methodology that is both usable and meets the needs of team members working on RBSPs. Neither all aspects of the software development nor management may be included in the study, and only the aspects that are addressed through interviews and questionnaire and those that are applicable will be considered. The strengths, weaknesses, and suitable areas of the common software development methods will not be investigated in this study. Those are taken from the current literature and detailed in the background information and literature review chapter. The limitations faced during this research study are listed in the concluding chapter, Chapter 6.

1.6 Structure of the Thesis

In Chapter 2, the background of the problem is stated alongside the related literature review. The research methodology, the steps, the data sets, and the details about the employed methodologies are described in Chapter 3. The analysis of the applied methods and the findings appear in Chapter 4. Later, Chapter 5 includes the proposed methodology in detail. The discussion of the methodology and expert opinions follow. Finally, Chapter 6 consists of the answers to the research questions, limitations of the proposed method, and the conclusion by summarizing the contributions and future works of the study.

CHAPTER 2

BACKGROUND INFORMATION AND LITERATURE REVIEW

2.1 Background Information

Before giving information about the current literature, and positioning our study in the literature, it is useful to provide some background information about various common software development methods since understanding their strengths and weaknesses and comparatively determining which ones are beneficial under what circumstances are essential to position our methodology.

The software development process occurs in various phases such as design, product management, and project management. There are different methodologies, namely Waterfall, Prototyping, Iterative, Incremental, Spiral, Rapid Application Development (RAD), Extreme Programming (XP), and Scrum. Each has their advantages and disadvantages [12]. The life-cycle model contains all the related stages of software development, including maintenance, and is considered as a subfield of systems development life-cycle.

Software development methodology is a framework that emerged in the 1960s based on the development life-cycle. These methodologies consist of a set of steps related to the day-to-day work of the team [13]. The flexibility of the frameworks allows the development team to introduce custom stages as needed to the development phase.

A system development methodology is a framework which is used for structuring, planning and controlling the process of developing an information system [14]. A wide variety of these frameworks have been introduced over the years, each with its own strengths and weaknesses, which implies that no one development methodology is ideal for all projects. Custom-tailored methods for specific development teams are the most usable ones.

The definitions of software development methodologies and their strengths and weaknesses are gathered from the literature including references [14], [15], [16], [17], [18], [12], [19], [20], [21], [22], [23], [24], and [25] as listed below.

2.1.1. **Waterfall** [19]**:**

The Waterfall is a linear development methodology with distinct steps. It is a formal method that utilizes a top-down development approach. Some of the phases can be merged, and the starting and ending points can be changed from project to project. The basic principles of the Waterfall are as follows:

1. The project is divided into sequential phases; minor overlaps are possible between the phases
2. Emphasis is heavy on planning, time schedules, target dates, budgets and implementation of the entire system at one time.
3. Control and monitoring are tight through the project life-cycle. Extensive documentation is done, and formal reviews and approval of the user are present.

The strengths of the Waterfall method include:

1. Ideal for less experienced teams and project managers.
2. Strict development phases let documenting and reviewing to be high quality, reliable and on time.
3. Progress is measurable.
4. The model is simple, making it easy to understand and use.
5. Resource allocation is pre-planned and easy to follow.

Some of the weaknesses of the Waterfall model are:

1. Rigid, well-defined phases convey inflexibility.
2. Production is slow and costly.
3. Backward movements are minimal, with progress always forward.
4. Little chance to use iteration.
5. The approach heavily relies on early identification and specification of the requirements, making it not feasible or applicable in some projects.

6. Requirement inconsistencies, missing components, and other problems are often detected at later stages, causing inconveniences.
7. Performance testing does not start before fully completing the system.
8. Once the testing stage is reached, it is challenging to return to a previous stage to fix specific problems.
9. Adapting to changes is difficult.
10. The approach is not suitable for long and ongoing software projects.
11. Excessive documenting is time-consuming.

2.1.2. **Prototyping** [20]**:**

Prototyping is an iterative process in which a working replica of the system is developed over time. It may be used within other SDMs. The basic principles of Prototyping are as follows:

1. It is not a complete stand-alone development methodology, but rather an approach to handle certain parts of the traditional methods.
2. It aims to reduce project risks by reducing the tasks into small manageable parts and developing them at early stages.
3. The user is involved in the development process.
4. Iterative modification occurs to the proposed prototypes to achieve the user's desired specifications.
5. Some prototypes are wholly discarded, while others can be kept and turned into actual products

The strengths of Prototyping can be listed as below:

1. It can model the essential aspects of the system in the early stages.
2. It improves user's participation and communication with the stakeholders.
3. It validates user requirements with the prototypes that are presented during the early stages.
4. Users can see and come into contact with the product.

5. It helps to identify missing functionalities or misunderstood functions of the system.
6. It improves the efficiency of the software product.
7. It encourages innovation and design flexibility.

The weaknesses of Prototyping include these:

1. The process is time-consuming and, hence, not suitable for a project that has a tight budget.
2. The designer should be experienced; otherwise, implementation difficulties can be incorrectly estimated, resulting in not finishing the deliverables on schedule.
3. It has no strict approval and control mechanisms.
4. Significant requirement changes may occur.
5. Non-functional items are hard to identify.
6. If the prototype does not include enough user needs, the resulting design may be inflexible.
7. If the prototype is thought to be a throwaway and is kept as the final product, the end-product quality would be lower than the acceptable level.
8. Multiple iterations of the prototyping phase negatively affect the budget and scheduling limits.

2.1.3. **Incremental** [21]**:**

The life-cycle of an Incremental model can be visualized as a Multi-Waterfall cycle. In the Incremental model, the development cycles are divided into smaller modules, thus making each module far more manageable. Each module passes through all the phases separately, including requirement gathering, design, development, and testing. The basic principles of the Incremental model are as follows:

1. Multiple Waterfalls are performed back-to-back in such a way that all the phases are followed, and one module is completed before starting the next one.

2. The overall requirements of the entire project may be well-defined before advancing the individual Waterfall phases.
3. As an alternative, all the requirements, the design and system core may be established and, later, individual parts may be completed through iterative Prototyping.

The strengths of the Incremental method are as follows:

1. Developers can achieve a moderate level of control with manageable module sizes. Related documentation can be prepared at certain milestones to evaluate the status properly.
2. Working software is created quickly at the early stages of the software life-cycle.
3. Stakeholders can see fully working modules as early which will ensure the delivery of the desired output.
4. It reduces integration and other architectural risks.
5. Each increment adds to the previously completed modules and forms more and more complete systems at later stages.
6. Debugging and testing during a small iteration is easy.
7. It offers a chance to monitor how the changes affect the system.

Some of the negative aspects of the Iterative model are as follows:

1. The overall system requirements may be neglected while handling multiple Waterfalls for individual modules.
2. Those modules to be integrated will be developed at different times, which means that the way they affect and interact with each other should be defined in the early stages.
3. Overlapping the phases of the iteration are not possible.
4. If difficult modules are left to later stages of the development, completing the entire system will be troublesome.

2.1.4. Spiral [22]:

The Spiral model focuses on project risks, and it is not a standalone development methodology. Usually, according to the nature of the risks, the Spiral model employs another traditional method within its life-cycle. The basic principles of the Spiral method are as follows:

1. Heavy focus on risk assessment and reduction. It divides the project into smaller parts to reduce development complexity and gives a chance to evaluate risks throughout the entire life-cycle of the project.
2. Each cycle guarantees specific steps to be followed repetitively. This is to ensure the necessary steps to reduce the risks taken through the project life-cycle.
3. Determining the objectives, alternatives, evaluating those alternatives and reducing risks, developing and testing the artifacts and planning the next iteration is part of each iteration of the Spiral model.
4. Each cycle gathers the needs of the stakeholders and, at the end of each iteration, decides whether they are met or not.

Strengths:

1. Heavy focus on the project risks reduces the chance of their occurrence.
2. It is possible to make changes to the project or add new functionalities at later stages of the development.
3. Spiral can adopt a second suitable method according to the risks associated with the project.
4. Customer feedback is welcomed at all stages of the project life-cycle.

The weaknesses that are identified with the Spiral method are listed below:

1. Choosing the correct set of methodologies for each cycle of the Spiral model can be challenging.
2. It is heavily tailored for individual projects and, since every project has a different set of risks, reusability is low in the Spiral model.

3. No strict deadlines are present in the iterations. Budget and scheduling issues may arise, potentially causing the chance of not meeting schedule or budget.
4. The Spiral model should be followed correctly for the smooth development of the project.
5. It is not suitable for smaller-size projects.
6. The cost of using the Spiral method is high.

2.1.5. Rapid Application Development (RAD) [23]:

RAD prioritizes prototyping above design specifications in order to develop the software system in a short amount of time. Compromises to the execution speed, usability and features may occur due to the short amount of time allocated to the project. The basic principles of RAD are as follows:

1. Fast development with low investment cost for a relatively high-quality product.
2. It breaks the project into smaller parts to reduce development complexity and risks that go with it.
3. It heavily utilizes computer-aided tools for fast development of quality systems.
4. Satisfying the business needs is foremost, whereas engineering and technically correct development are secondary.
5. Strict delivery deadlines exist. Usually, the features are dropped to meet the deadlines instead of extending the deadline itself.
6. User involvement is necessary to identify the needs and verify the product promptly.
7. RAD does not employ throwaway prototypes. Instead, the prototype developed through the life-cycle is converted to the actual product.

Some of the strengths of RAD are as seen below:

1. The earlier stages of the development operational version of the product are present.
2. Developing the software quickly usually reduces production costs.

3. RAD reduces development time.
4. The user's opinion is considered while focusing on the essential system elements.
5. User demands can be quickly integrated into the system.
6. Reusability of the components is increased.
7. The approach significantly saves budget, production time and effort.

The downsides of the RAD are as follows:

1. Increasing the development speed and lowering the production cost may result in lower quality products.
2. Highly experienced teams and individuals are needed to set the requirements correctly.
3. If there is missing information between the user and the developers, the developed system may diverge from the expected product easily without any control mechanism.
4. More requirements than needed may be introduced, causing unnecessary work to be done at later stages.
5. The resulting product may be loaded with an unnecessary number of extra features.
6. Since documentation is not a priority, coding and other practices may be violated during project development.
7. The resulting systems usually are not modular. They are not easily reused.
8. RAD is not suitable for smaller projects with limited budgets as the costs for modeling and code automation is high in RAD.
9. Due to the nature of rapid development, scalability is not a primary concern, only stated business needs are. The resulting product may not be scalable.
10. To be able to integrate modules developed on different Prototyping iterations, the interfaces should be defined well and early in the project life-cycle.

2.1.6. Extreme Programming (XP) [24]:

Extreme Programming is an agile software development methodology developed by Kent Beck in 1999. Beck's vision was that to write any code; first, tests of that code should be written. Also, coding should be done in pairs where one person does the actual coding while the second person generates further ideas. His goal was writing high-quality software quickly while adapting to changing requirements. XP has five main values. These are namely: communication, simplicity, feedback, courage, and respect [26]. Through these values; planning, analyzing and designing phases are not done for the whole project. Instead, they are performed little by little when needed, thereby helping to reduce the cost of changing software.

In this study, two examples from the agile methodology, namely XP and Scrum have been explained. These are two of the most common agile methods [27]. In Chapter 5, Agile manifesto, values and principles of agile are also listed.

The basic principles of XP are as follows:

1. The focus is on quickly realizing the actual code rather than detailed analysis, planning, and modeling at the beginning.
2. The features are called "stories" in XP.
3. The customer picks the next release with the most valuable stories among all of them.
4. The stories are turned into smaller tasks by the programmers, who take responsibility for development for several tasks.
5. The programmer prepares test cases to show the completion of the tasks.
6. Working with another person, the programmer completes the assigned tasks for release.
7. There is a set of XP practices utilized throughout all the project phases. These are user stories, small releases, metaphor, simple design, tests, refactoring, pair programming, continuous integration, collective ownership, on-site customer, 40-hour weeks, open workspace, and just rules [24].

Strengths:

1. Allows companies to save time and cost to realize the entire project. Less documentation and more focus on the deliverable product enable cost reduction. Problems are usually solved by communication among team members.
2. XP projects maintain simplicity. The code is somewhat simpler and can be improved at any moment that is necessary.
3. The development process is visible and accountable for the use of tests. Also, developers can easily show progress.
4. End-user involvement is high, and there is constant feedback because of it. This is necessary to identify and make fast changes.
5. Employee satisfaction and retention is usually high in projects that follow XP

Weaknesses:

1. XP tends to focus on code more than design. Since good design is essential to develop sound software products, this may be perceived as the main disadvantage of using XP.
2. Quality assurance is not measured in XP projects. As a result, defects may occur in the code.
3. XP is not a suitable methodology for the projects where the programmers are separated geographically, or the customer is not able to be present with the development team.
4. Lack of documentation requires the developers to be more experienced. Since they are forced to solve most problems among themselves, a good understanding of the underlying system and the technics is, therefore, required for the team.

2.1.7. **Scrum** [25]**:**

In [28], the definition of Scrum is given as follows: “Scrum is a framework within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value. Scrum itself is a simple framework for effective team collaboration on complex products.” Scrum has been

used for different types of projects. Because of this, it is known as an agile project management tool. Scrum is designed mainly for small teams of less than ten people in which the work is broken into parts that can be completed within specified periods, called ‘sprints.’ The duration of sprints may vary from two weeks to a month.

Basic principles of Scrum:

1. Scrum has three leading roles. Product Owner, the Scrum Master, and the team. The Product Owner represents the client organization and is an active member of the team. The Scrum Master is the project manager that measures and controls the process. The team is the people that develop the project as expected.
2. The Product Owner creates a backlog which are the requirements ordered by their priorities.
3. The sprint planning team takes the backlog and selects a few priority items to complete within the current sprint.
4. Daily stand-in meetings which are less than fifteen minutes are held by the Scrum Master which allow him to track the progress and guide the team.
5. A review of the completed sprint is made, and the next sprint is started.
6. The entire process repeats itself if there are items in the backlog that are not solved yet.

Some of the strengths of the Scrum are listed below:

1. The Scrum method helps developers save money and time.
2. It requires little documentation. Hence, in projects where business requirements documentation is hard to form can be developed using Scrum.
3. Since the sprints are for short durations and a set of backlog items are handled at any given time, fast development is possible and can be tested quickly with Scrum.
4. Regular meetings are held which increase the frequency of progress updates.
5. Project development can be tracked easily.

6. Scrum is iterative and continuous feedback is taken from the user.
7. Short sprints, a small number of backlog items and continuous customer integration makes it easier to make changes in projects.
8. Daily meetings increase individual productivity and make identifying problems easy and fast.

The weaknesses of the Scrum are as follows:

1. If there is no definite end date of the project, there is a chance that the customer will continuously demand new functionalities to be added.
2. For projects where, specific tasks are not well defined, estimating the time and budget costs can be difficult. Also, some of the tasks may be done over multiple sprints, causing inaccurate cost estimations.
3. Team members need to be both experienced and committed. Otherwise, the project will suffer extensively.
4. For larger teams, coordination and fast movement is a challenge, rendering Scrum inapplicable.
5. Novice individuals cannot be part of the team. All members need to be experts in their areas. If some members leave during the project, it will have a sizable adverse effect on the project.
6. Quality management is hard to employ in Scrum projects.

2.1.8. Comparison of Software Development Methodologies

A comparison of various software development methodologies is made in studies such as [16], [29], [30] and [31]. In [16], they have identified different project characteristics that influence the decision of selecting the most appropriate software development methodology for a specific project. Tables 2.1 – 2.4 shows these different project characteristics of the common SDMs based on requirements analysis, the status of the development team, user participation, project type, and associated risks. Accordingly, no one methodology is suitable for all types of

software projects. All the tables ranging from Table 2.1 to Table 2.4 are modified by the authors of this study to incorporate the comparison of the Scrum methodology.

Table 2.1 – Characteristics of SDMs Based on Requirements Analysis:

Requirements analysis	Waterfall	Prototype	Iterative	Spiral	RAD	Agile	
						XP	Scrum
Are requirements easily understandable and defined?	Yes	No	No	No	Yes	No	No
Do we change requirements quite often?	No	Yes	No	Yes	No	Yes	Yes
Can we define requirements at the starting of iteration?	Yes	No	Yes	No	Yes	No	No

Table 2.2 - Characteristics of SDMs Based on Status of Development Team:

Development Team	Waterfall	Prototype	Iterative	Spiral	RAD	Agile	
						XP	Scrum
Less experience on similar projects	No	Yes	No	Yes	No	No	Yes
Less domain knowledge (new to the technology)	Yes	No	Yes	Yes	No	No	No
Less experience on tools to be used	Yes	No	No	Yes	No	No	No
Availability of training if required	No	No	Yes	No	Yes	Yes	Yes

Table 2.3 - Characteristics of SDMs Based on User's Participations:

User's participation	Waterfall	Prototype	Iterative	Spiral	RAD	Agile	
						XP	Scrum
User participation in all phases	No	Yes	No	No	Yes	Yes	Yes
Limited user participation	Yes	No	Yes	Yes	No	No	No
User has no previous experience of participation in similar projects	No	Yes	Yes	Yes	No	No	No
Users are experts of problem domain	No	Yes	Yes	No	Yes	Yes	Yes

Table 2.4 - Characteristics of SDMs Based on Project Type and Associated Risk:

Project Type and Risk	Waterfall	Prototype	Iterative	Spiral	RAD	Agile	
						XP	Scrum
Project is the enhancement of the existing system	No	No	Yes	No	Yes	Yes	Yes
Funding is suitable for the project	Yes	Yes	No	No	Yes	No	Yes
High reliability requirements	No	No	Yes	Yes	No	Yes	Yes
Tight project schedule	No	Yes	Yes	Yes	Yes	No	Yes
Use of reusable components	No	Yes	No	Yes	Yes	Yes	Yes
Are resources (time, money, people, etc.) scarce	No	Yes	No	Yes	No	No	Yes

Both the traditional software development methods and agile ones fail to fit the general process of research-based software development. It is clear to see that traditional methods, such as Waterfall and RAD, call for the requirements to be set at early stages. Once done, these requirements should not change further to complete the development successfully. This is not possible in RBSPs which at times employ inexperienced graduate students within the development team, unlike agile, prototype, and iterative as they require expertise in the entire team. Also, in agile methods, team members are expected to solve problems among themselves rather than relying on extensive documentation. These are just some of the examples showing the incompatibility of the common SDM for RBSPs. Set of incompatibilities of SDMs are given in Chapter 5 as a table.

The authors have collected the needs and the problems of RBSPs via interviews and questionnaires, which will be discussed in the forthcoming chapters. However, the items that are described in Tables 2.1 – 2.4 in order, can be generalized for RBSPs as below:

1. Since research-based projects heavily rely on the research component, not all requirements may be apparent at the beginning of the project and may be discovered during the project life-cycle.
2. Requirements may often change with the discovery of ideas and methods during the project development.
3. These types of projects are usually funded by national or international funding agencies, which often demand that researchers list the requirements before approval. As a result, an abstract idea of the requirements should be set followed by changes for subsequent acceptance.
4. Research-based projects generally require a complex system to be built due to difficulties related to research methodology, underlying science, or other issues and difficulties.
5. Often, the development team in such projects comprise graduate students and researchers. Although researchers and professors are regarded as more experienced on similar projects, have enough domain knowledge, and possess

experience with the development tools, graduate students usually are new to the field and lack due background.

6. Training of scholarship students is required in some cases which need to be considered in project planning.
7. In RBSPs, there may be either no participation or full participation by end-users, which matter profoundly depends on the nature of the project and how it is initiated.
8. End-users may be experts in the field and be or not be experienced with similar projects.
9. RBSPs usually include designing a novel system with significant research involvement. However, some projects are funded by funding agencies which let researchers enhance a previously generated system.
10. RBSPs usually require funding to be carried out.
11. In most cases, high-reliability systems are developed by research-based projects, be it the realization of a mathematical model or cutting-edge technological software.
12. Research-based software projects have tight schedules because of the funding agencies. In most cases, again funding is for a fixed period, and one must show satisfactory progress through the development phase. At the end of the funding period, usually, no extension is granted by these agencies.
13. Resources can be considered scarce. Funding agencies insist on giving the amount that will satisfy the project needs. Generally, in order to be funded by these agencies, academicians ask for reasonable amounts of resources and no more.
14. Overall, the reusability of the components is not the primary concern of such projects. However, if the project is expected to be expanded after its completion or similar projects are expected to be carried, then reusability may be a concern.

2.2 Literature Review

Creating a successful software project is not an easy task. Both industry and universities struggle to develop successful projects from time to time. In [32], the authors conduct a survey of over 200 middle-level to senior programmers to identify the percentage of IT software project failures. The findings reveal that around 11% - 15% of such projects are terminated even before any deliverables are completed due to changes in requirements and the project scope as being the main reasons. Also, around 16% - 22% of the completed projects are considered unsuccessful.

In the literature, various studies show the problems faced in software projects. In [33], low qualification of the employees, lack of clear goals, inadequate financial planning of the project, insufficient attention from project heads and unstable legislation are listed as some of the reasons behind these problems. In [34], lack of cultural understanding in project teams and lack of communication are considered as some of the main barriers. Communication problems are more apparent in Global Software Development (GSD) initiatives rather than the standard ones since team members usually consist of people from around the world, making coordination and communication with team members more critical cases. In [35], the authors propose to use process patterns to detect coordination problems in the project team. These patterns contain information about potential violations but using them is labor-intensive for project managers. Identifying coordination problems becomes more difficult when multiple staff is responsible for various tasks and when tasks often change in a dynamic environment. Although the present work attempts to include some of the best practices and problems of the industry as part of its solution, it does not include software projects that are carried out by the industry explicitly.

University and industry software projects are not always wholly separate, and there are some collaborative studies made between the two sectors. Both parties gain from a possible collaboration [36]. The industry gains more relevant research results and researchers can better understand the scope and type of problems to attack. In the present study, a series of sample works by both industry and academicians is considered, and lessons learned from such interactions are reported. As a way to increase these collaborations, academic researchers and students need to talk to the companies to learn from their real problems, while industry practitioners are

expected to find researchers of the field as they are likely to have more detailed and up-to-date data about their problems. Collaboration between industry and universities is commonly consultancy-based. In [37], a different type of collaboration is also listed. Payment of software professionals in empirical studies, payment of software companies for extra work, the researcher acting as a client, use of software professionals in industry seminars and short experiments and surveys can also be considered as other forms of industry-university collaboration. Although such partnerships with the industry are essential, this study mainly focuses on the software projects that are research-based and completed by universities.

Some alternative methods can be used instead of common SDMs during software development in universities. For example, the authors of [38] have proposed the use of Evidence-Based Software Engineering (EBSE) for projects as rooted in Evidence-Based Medicine (EBM), whose goal is "the integration of the best research evidence with clinical expertise and patient values" [39]. The EBSE method is said to be suitable for research-based projects since it involves multiple possible solutions to a problem to be tried and the best solution to be picked at the end. However, it can be costly when applied to the entirety of the project artifacts, but still useful for project parts that are critical. Furthermore, the solution is unknown at the beginning because of the research-heavy requirement of the parts. EBSE is conducted in five steps: (i) Convert the need for information into answerable questions; (ii) Track down the best evidence that answers the questions at hand; (iii) Critically appraise the evidence for its validity, impact, and applicability; (iv) Integrate software expertise and stakeholders' values into the critical appraise; and (v) Evaluate own effectiveness in steps one through four and try to improve them next time.

Open Source Project Management is another method that is employed by geographically separated people to work together and develop a software system collectively. Researchers have investigated why talented developers contribute to Open Source Software (OSS) without any monetary incentives [40]. In [41], it is shown that most OSSs are developed by a single developer. Although there are multiple contributors to the project, in the beginning, most fail to commit to the project extensively. There is only a handful of successful OSS projects where many

developers get together to work. Apart from this, OSS is a loose development methodology which does not work well within allocated time and budget limits.

There are limited studies on research-based software projects. However, the existing ones clearly show the gap in the literature and explain the need for a new methodology that is tailored to the needs of the researchers working on such projects. In this respect, researchers of [2] investigate the need for a novel software development methodology for software projects in universities in the form of a case study focusing on four projects initiated by universities, including one from Sri-Lanka. Project management is said to be mainly dependent on the industry. As said before, no one method is suitable for all the projects and different methods are utilized in different situations. No method focuses on research, innovation, and learning at the same time. Hence, the authors of [2] suggest that there is a need for such a methodology. In the study, researchers focus on the adaptation of agile project management to manage IT projects in Sri-Lanka, with the finding that adaptation is around 25%, which is insufficient. In the same study, successful software projects around the world that are initiated at universities are considered. Namely Google, Linux operating system, Apache, and Vidisayura, which is a successful project from Sri-Lanka, are the focus of the case study. The authors conclude that creating a novel software development methodology for such projects at universities is a hard task because of the innovative and dynamic structure of such projects. It is additionally observed that, in the early phases of the project development, none of the projects mentioned above use common software development practices.

In [5], the differences between industry and universities in software development is discussed. The researchers mention entrepreneurial universities, which form institutions like firms that develop software products. The study states that these organizations rarely adopt common software development methodologies in their development cycles. One of the main differences between them and the industry is observed as in the motivation section; the industry mainly focuses on software development, and research is a secondary issue. At universities, however, the situation is reversed. The primary focus is on research and to do so; software development is carried. The expected outputs are also different; academics focus on research findings and publications whereas the industry mainly focuses on delivering

products and making a profit. Including the end-user in university, projects are often hard because it is considered to be a voluntary act. In the industry, though, there is a client and stakeholders that pay for the creation of the project, so they are more invested in the project itself. Another point is that university researchers are more open to trying out new ideas and changing the system architecture in future stages; whereas the industry is bound with the contract at the beginning of the project and does not tend to diverge from the initial planning. Lastly, researching a novel software development methodology for universities is listed as one of the future works of studies by these institutions.

In [4], an analysis is made of the factors contributing to software development at universities. From the literature, the researchers divide the projects into three stages as project initiation, system development, and implementation. In each stage, they list a set of factors contributing to software development. During the project initiation stage, the topic, issues, concerns, community needs, resources, time, staff, funds, and area experts are found as some of the components that contribute to the success of a project. During system development, skilled staff, ideas, objects, and team structure are the main factors. During implementation, immediate user adoption, intellectual property licensing, finance, skilled staff, communication networks, technology transfer offices, marketing activities, communication among members, industry-university partnerships, and start-up firms are found to be the factors. These findings show a more compact list of factors for the three stages. For project initiation: *problem identification*, *team*, *interested community*, for system development: *team*, *interested community*, *development skills, and resources*, and finally for Implementation: *team*, *interested community* and *sustainability* are listed as the significant aspects affecting the success of software development. In conclusion, researchers state that these factors can be used to propose a suitable software development methodology for entrepreneurial universities.

In [42], a thesis study is conducted where differences between industry projects and academical projects are considered. The Grounded Theory method is used to collect and analyze the needed information through interviews, forced ranking and a case study. An analysis of the data shows significant differences between industry and academic software projects. *People retention, lack of funds and economic*

uncertainty, *access restriction to publication libraries*, *late plans on project development*, *getting end-user interaction*, *problems due to industry partnership*, *problems with funding agencies*, and *product maintenance issues* raised by the user are found to be the main challenges faced by the academicians that are involved with research-based software development at universities. Also, best practices are categorized and listed for research supervisors, research groups, universities, and regulatory institutions individually. At the end of the study, a framework is proposed based on the considerations of each group. In the conclusion of the study, researchers state that there is currently no software development methodology to respond to the needs of individuals working on research-based software development projects, adding that additional studies can be pursued to propose a new software development methodology. These findings also precisely point to the gap in the literature and show the necessity for the present research and its motivation.

CHAPTER 3

RESEARCH DESIGN & METHODOLOGY

This chapter explains how the present work is formed and which research steps are taken to complete it. All the selected methods will be explained alongside all the necessary steps of those methods to gather and analyze the data. Through the literature, it is observed that the topic of research-based software engineering is rarely delved into by researchers. In the literature review, it is already shown that for Software Engineering research that heavily involves a human factor, qualitative research methods are widely used and accepted. Here, since we gather the needs of the academicians and their best practices, human involvement is also very high. For this reason, we use qualitative research techniques to identify the problem and gather data to form the necessary solution to the stated problem. For this purpose, various qualitative research methods are considered among which, one is selected as suitable as explained herein. All the qualitative research methods fall under two types of reasoning approach: inductive and deductive [43]. Deductive reasoning utilizes a top-down approach and takes a given theory about the topic of interest and tests that theory to narrow it down to a more specific one. Inductive reasoning works, though, in the opposite direction; it is considered as a bottom-up approach, and the researcher starts by detecting patterns and relationships among concepts and, then, proceeds to form a hypothesis using them.

3.1 Alternative qualitative research methods

The other methods considered for this study are Action Research, Ethnography, Case-Study Research, and Grounded Theory. Action research (AR) [44] is a research process in which both the researcher and the area expert actively participate. The inquiry is conducted by and for those taking action. AR consists of a series of processes [45] such as selecting a focus, clarifying theories, identifying research questions, collecting data, analyzing data, reporting results, and finally taking informed action. These steps cyclically repeat themselves until the desired improvement level is achieved. Action research is mainly used to test theories and

not to form them. Ethnography [46] studies the social interactions of the people that are in their natural environment and offers insight into a person's views and actions. To derive meaningful results, the researcher would need to stay among the research subjects for an extended period to actively or passively observe them. These types of research are useful when the researcher intends to observe the natural environment of the subjects without changing the settings of their environment. Case-Study Research (CSR) [47] is an in-depth study of a situation rather than statistically surveying a large set of happenings. CSR does not offer a full answer to a situation because of its scope restrictions, but still correctly directs the researcher to create a hypothesis. Grounded Theory (GT) [10] is a research method which lets the researcher categorize and conceptualize the patterns and structures of the area of interest through constant comparison. At first, codes are generated from the data which are later compared with each other to reveal the links and relations among them. Finally, theories are formed accordingly.

3.2 Research Design

Previous studies such as [42] have utilized the seven questions that are asked in [48] to make suitable choices for their qualitative research designs. Similarly, we answer the following seven questions to make educated decisions about our research design.

1. What types of information or data are collected?
   The types of software development practices the individuals in RBSPs use, the issues they face, and their best practices are collected. To this end, the data related to people, interactions, actions, limitations, rules, goals, structure, and practices are gathered.
2. What are the data collection techniques?
   Semi-structured interviews, questionnaires, expert opinions, memos, documents related to research-based software projects and literature review.
3. Where and among what group or groups of people is the research conducted?
   The study will mainly take place at universities in Turkey where individuals are involved in funded RBSPs. The subjects to interview are selected by their availability, relevance, and accessibility. Convenience sampling [49], which is a non-probability sampling technique in which participants are selected due to their

convenient accessibility and their proximity to the researcher, is used to reach out to various researchers for the study.

4. What strategies are used for data triangulation?

   The datasets collected from the universities will be compared with the findings from the literature. Semi-structured interviews will be done with more than a person per project when available to increase the rigor of the study.

5. Is the study undertaken by one person or with the assistant of others? Is there multiple investigator triangulation?

   All the data will be collected by the author of the study. However, the analysis will be done under the guidance of the supervisor; that is, two individuals will conduct the study. In this way, triangulation will be ensured.

6. What are the theories framing the present study (theoretical triangulation)?

   The common software development methods as described in Chapter 2, qualitative research practices, software engineering practices, and issues and best practices taken from the literature in Chapter 2 are used extensively to form this study.

7. Is the study funded? How much will the project cost in time and money?

   The project is not funded. This is a Ph.D. thesis study. As for the timeframe, the thesis alone is expected to take less than four years.

In this study, the authors aim to come up with new theories that will demonstrate both the needs and the best practices of the individuals that work on RBSPs making it inductive research. Ethnography and Action Research call for the researcher to be involved in the action or observe entire processes, neither of which is practical or handleable in the stated timeframe. As mentioned before, the CSR is mostly deductive in which one tests their previous hypothesis on a selected sample. In GT, the researcher generates codes, find relations among them and can reach a hypothesis in a bottom-up approach. As far as the needs and the benefits of the research methods go, GT proves to be the most suitable candidate.

The problems, best-practices, and factors affecting the success of university projects are collected in Chapter 2 as per the literature in [42] and [4]. The findings of these studies are considered while proposing a new methodology for RBSPs in Chapter 5. Theoretic triangulation is ensured with the inclusion of these previous findings.

3.3 Data Collection

Both primary and secondary data collection is done in this study. Secondary data collection relies on the results of the previously conducted similar studies. The findings of the previous studies will also be taken into consideration while proposing a new methodology as explained above.

In this study, three types of primary data is collected. These are interviews, questionnaires, and expert opinions.

3.3.1 Interviews

Interviews are done with twenty individuals who have worked on fifteen different research-based software projects. All the projects were originated from the universities and had external funding from either a national or international funding organization. The software projects are numbered from one to fifteen, and the interviewees are numbered from one to twenty to anonymize the participants. Convenience sampling is utilized while selecting the participants for the interviews as explained before. Although there is no personal information taken from the interviewees and it is envisioned that the questions asked are not disturbing, and moreover it has the option of not participating in the study or leaving at any time, a letter of consent is taken from all the attendees. Besides, to be sure, the researcher applied to the Ethics Committee of the University and received approval for both the interviews and the questionnaire.

Table 3.1 below lists the necessary details of all the projects and the related interviewees. Following Table 3.1, the projects are explained in written form. Detailed information about the projects as well as the analysis of the interviews is given in the sequel.

Table 3.1 - List of projects participated by interviewees

Project No	*Project Area*	*Interviewee Numbers*	*Interviewee Positions*	*Funding*	*Staff*
1	Designing and simulating a drilling device	1	Researcher	Domestic, 2 years	2 graduate students, 4 researchers, 1 project manager
2	Health informatics curriculum design	2,8	Researcher, Graduate Student	International, 2 years	3 graduate students, 5 researchers, 1 project manager
3	Designing and building an oil rig	3, 9	Researcher, Graduate Student	Domestic, 2 years	2 graduate students, 4 researchers, 1 project manager
4	Developing a security solution for governmental institutions	4	Researcher	Domestic, 2.5 years	5 graduate students, 4 researchers, 1 project manager
5	Surgery simulation	5,12	Researcher, Graduate Student	Domestic, 3 years	4 graduate students, 5 researchers, 1 project manager
6	Remote access laboratory	6, 10	Project Manager, Researcher	International, 2 years	5 graduate students, 14 researchers, 1 project manager
7	Jet Engine Design	7	Project Manager	Domestic, 3 years	1 project manager, 3 graduate students
8	3D surgical locator	11,13	Researcher, Graduate Student	Domestic, 3 years	1 project manager, 5 researchers, 7 graduate students
9	Dyslexia Detection	14	Graduate Student	Domestic, 2 years	1 project manager, 2 graduate students
10	Virtual and face to face education design	15	Project Manager	International, 2 years	1 project manager, 5 researchers, 10 graduate students, 30 testers
11	Natural Language Processing for the Turkish Language	16	Project Manager	Domestic, 1.5 years	1 project manager, 1 researcher, 3 graduate students
12	Technological Teaching Material for Mentally Disabled Students	17	Graduate Student	Domestic, 1.5 years	1 project manager, 3 researchers, 2 graduate students
13	Innovative Mathematical Applications for Classrooms	18	Researcher	International, 2 years	1 project manager, 9 researchers, 5 graduate students
14	Online Learning Material	19	Researcher	Domestic, 2 years	1 project manager, 1 researcher, 3 graduate students
15	Sensory Networks	20	Researcher	Domestic, 1.5 years	1 project manager, 4 researchers, 5 graduate students

Project 1 is about designing and simulating a drilling device and heavily software-focused but still multidisciplinary. The funding was from a domestic organization and for two years. Two graduate students, four researchers, and a project manager were present in the project. At the end of the funding period, the project was deemed successful. The interviewee was a researcher in the project.

The second project is related to health informatics curriculum design. The software design was a part of the overall project. Funding was given by an international

organization for two years. The staff consists of three graduate students, five researchers, and a project manager. The first interviewee is a researcher and the second one a graduate student. The project still has six months of funding, thus still incomplete.

Project 3 includes designing and building an oil rig, involving both software and hardware components. It was funded by a domestic funding organization for two years as an ongoing project, where two graduate students, four researchers, and a manager are employed. The first interviewee is one of the researchers and the second interviewee is a graduate student.

Project 4 is related to developing a security solution for governmental institutions. The project includes security software development and the deployment of a fully functioning security system — a local agency funded the project for two and a half years. Overall, nine people work on the project: five graduate students, four researchers, and a project manager. The project succeeded at the end of the funding period, and the interviewee is a researcher in the project.

The fifth project is a simulation of brain surgery. Software components come together with haptic devices and other hardware components to increase the authenticity of the simulation. The project was funded by a domestic funding organization for three years. A total of four graduate students, five researchers, and a project manager were involved. At the end of the funding period, the project was successful. The interviews were done with a researcher and a graduate student.

Project 6 develops a remote access laboratory by combining software and lab device hardware for the purpose. The funding was taken from an international organization. In the project, five graduate students, one project manager, and fourteen researchers from six countries participated with some as customers involved in only some parts of the project. At the end of the funding period, this project was also deemed successful, maintained and utilized for more than three years after the funding period. The interviewees were the project manager and a graduate student.

The seventh project was about designing a jet engine. The software component of the project heavily relies on mathematical calculations and precise engineering. A domestic agency gave the funding for three years. Only a single project manager and

three graduate students participated in the development. The project was deemed successful, and the interview is conducted with the project manager.

Project 8 was related to a 3D navigation tool for a specific surgery operation. A domestic funding agency funded the project, and both medical and computational experts participated in the project overseen by one manager, and five researchers and seven graduate students as participants. The project was completed successfully. The first interviewee was a researcher in the project and the second one a graduate student.

Project 9 was about developing software that can detect dyslexia in students. The project is funded by a domestic funding organization for two years. Only one project manager and two graduate students participated in the development of the project. As of writing this thesis, the project is on hiatus, and it is not clear whether the team will be able to complete it. The interview was conducted with one of the graduate students in the project.

In Project 10, a virtual and face-to-face education system is designed and developed. There was a single project manager, five researchers, ten graduate students and 30 testers with funding from an international agency for two years. Three countries cooperated in the project, which is still ongoing. The interview is done with the project manager.

Project 11 tackles the problem of developing a natural language processing software for the Turkish language. The funding is taken from a domestic agency for 36 months, and the team is composed of a project manager, one researcher, and three graduate students. The project ended successfully and maintained by the team members. As a result; the application is still being used today. The interview was conducted with the manager of the project.

Technological teaching material for mentally disabled students was developed in Project 12 comprising a project manager, three researchers, and two graduate students and funded locally for 36 months. Upon termination of the funding, the project was complete. The interview is done with one of the graduate students of the project.

In Project 13, innovative mathematical applications are developed for classroom environments. The project was funded by an international institution for two years and conducted by three countries with a single project manager, nine researchers, and five graduate students. It is an ongoing project and, by now, six months is left for it to be completed. The interview is conducted with one of the researchers in the project.

Project 14 was to develop an online learning material interface and financed locally for two years. There were one project manager, one researcher and three graduate students in the project as well as third-party developers as additional support. The project was completed successfully. The interview is done with the researcher of the project.

The last project, Project 15 was about sensory networks and funded by a domestic agent for 36 months. In the project, there was one manager, four researchers, and five graduate students, who completed the work successfully. The interview is conducted with one of the researchers within the project team.

The conducted interviews were all semi-structured consisting of thirty-four questions addressing the issues and challenges faced during the development of RBSPs and the best practices which help develop successful projects.

A full list of interview questions can be found in Appendix A in Turkish and Appendix D in English. Here is a glimpse into some of the items:

- Did you realize a traditional software development methodology in your project?
- How many full-time and part-time staff do you have?
- What was the primary motivation of the staff to join the project?
- How often did you conduct meetings?
- Did the graduate students possess enough background? Did their skills match the project expectations?
- Did the academic hierarchy that the staff is in negatively affect the project development?

- Was there an end-user at the inception of the project, or did the research idea originate from the researchers? If so, how involved were the end-users?
- Did any unexpected risk appear during the development of the project? What were the reasons?
- Did you notice any communication problems among the project staff? What were the reasons behind them?
- How much do you think traditional software development methodologies apply to research-based university projects?
- What are the differences between industrial projects and research-based projects?

When to stop collecting data through interviews is an important issue to be considered in qualitative research studies [50]. A widely accepted method is to check whether one has reached saturation or not [51]. The saturation indicates whether, considering the collected data and its analysis, further data collection and analysis is unnecessary. Two types of saturation methods have been used in this work, namely Theoretical Saturation and Data Saturation. According to [9], grouping is done to categorize the collected data. If conducting new interviews stop adding any more useful information which contributes to previous categories, theoretical saturation is reached. Data saturation is achieved when no additional coding is being generated through data collection, and further coding is not feasible [52]. In this study, interviews are concluded when the analysis yields no additional theories and codes in the last couple of interviews. In the same way, before the analysis, the authors noticed a rapid increase in the repetitive answers in the final interviews, which is considered as another indicator that saturation is being reached.

3.3.2 Questionnaires

The Questionnaire to validate the findings of the interviews is shared with peers to see whether the questions are understandable and related to the findings of the interviews. After receiving feedback from the experts and revising questions accordingly, the questionnaire is distributed among 47 participants of an international conference on computer science and engineering with a specific track for software

engineering. Among the participants, 40% stated that they have been working as an academician for 0-5 years, 26% between 6-10 years and the remaining 34% more than ten years. Only 2% of the participants stated that they do not know about software engineering and project management; 4% stated they have little knowledge, 38% stated they have moderate knowledge, 38% stated they have good knowledge, and the remaining 17% stated they have very good knowledge about software engineering and project management. When asked about how many research-based software development projects that the participants had been involved in as a PM, researcher or a graduate student they answered: 17% never participated, 19% only participated in a single project, 36% in two to three projects, 11% in four to five projects, and the remaining 17% in more than 5 research-based software development projects. The questionnaire can be found in Appendix B in Turkish and Appendix E in English. The analysis of the questionnaire is given in the following chapter along with the analysis of the interviews.

3.3.3 Expert Opinions

After conducting all the interviews and analyzing them, a suitable questionnaire is prepared to validate the findings. The results of the interviews, questionnaires and the findings from the literature are used to propose a new software development methodology for research-based software projects. The proposed methodology is evaluated by using the expert opinion method. Due to time constraints, CSR is not an available choice to evaluate the proposed methodology. In the absence of empirical data, expert knowledge is regarded as the best or only possible source of information [53] [54]. Experts can be identified by their qualifications, training, experience, professional memberships, and peer recognition [55]. In the present two types of expert definition is done: (i) Academicians at the level of Assistant Professor or higher, working in a university department related with software engineering for more than five years and participated in RBSPs. (ii) Academicians that are at the level of Assistant Professor or higher, working in any university department for more than five years and participated in more than five RBSPs. With the inclusion of these two types of experts in the study, the expertise from both Software Engineering, and RBSPs are covered. Expert opinions are gathered from eight experts; seven were academicians for more than ten years and one for five to ten years. Five of the

experts claimed to have average knowledge about software development methodologies; one claimed to have very good knowledge, one with good knowledge, and the remaining one little knowledge about SDMs. From all the experts, three had participated in more than five RBSPs, four in 2-3 RBSPs and one in just one RBSP. Two of the experts that had participated in more than five projects were from a department that is not related to computer science or software engineering. The analysis of the expert opinions is exhibited as the last item of Chapter 5.

3.4 Implementation of Research Methodology

To come up with a sound methodology, the research problem is studied and discussed among the researchers and their available peers. Later, an extensive literature review is performed to position the research topic accurately. Possible research methodologies are discussed, and a suitable one is selected.

Software engineering significantly involves human components. To do research that can capture the nature of the human element, qualitative studies are suitable for this field. As explained before, in this study, multiple qualitative research methods are considered. Different methods have different qualities. Here, as stated previously, Grounded Theory [9] is found to be the most suitable option.

Grounded theory was first proposed by Glasser & Strauss 1967 in their book named "The Discovery of Grounded Theory" [9]. Since then, there have been many variations of how GT research should be conducted as a systematic way of discovering a theory through data collected using social research [56]. Conducting interviews, questionnaires and taking field notes are standard practices to collect data to analyze. GT is also an inductive method that utilizes the data to the extent of forming theories [10], which implies that one collects the data, analyzes it individually and compares them against each other to derive meaningful relationships amongst them. All of this, in the end, should result in forming a complete theory about the research question. When GT was initially formed, only a single theory could be derived from the collected data. Later on, with the new variations of GT, multiple theory forming is made possible [57]. Every variation of GT has some specific steps to follow. The original pattern started with open coding,

which takes the transcribed interview or any other type of data and labels them. Giving labels related to the content of the data is called 'coding' in GT which can be done manually by traversing the data, line by line, or automatized with modern qualitative research tools such as NVivo [58]. After open coding is performed, the relations among the codes need to be discovered. This is done in the axial coding phase, where the codes are grouped by their relations regardless of physical actions, contextual similarity, or people related to them. These relations generate specific and separate groups in the codes, which will, later, be used to generate theories. Constant comparison is required with other groups to make sure that the groups that are formed are logical and sound. The final step of the research is integrating multiple categories, refining them and forming actual theories.

In this study, the GT variation proposed by Strauss and Corbin [57] is used to enable the integration of the literature study as a base point for research to be done as well as to make room for multiple theories to be formed at the end of the study. Different from the original GT, the steps of the GT in [57] include open coding, axial coding, and selective coding.

3.4.1 Open Coding

In this study, open coding is performed to structure the data into categories. This takes long lines of textual data and makes an abstraction of its contents. Open coding is performed by the researchers considering the actions taken in the projects, types of problems and challenges faced, and the type of best practices carried out by them. For example, about the experience of the graduate students in the project, Interviewee 4 stated "Technically speaking, graduate students were sufficient. Security-wise, their background was not exhaustive. Researchers mostly had to train and teach graduate students as they would be the ones to develop the system. Also, directing the graduate students and monitoring them closely increased the success rate of the developed modules." This text is coded with open coding as: "*Sufficient experience*," "*Training done*," and "*Close monitoring*." "*Sufficient experience"* shows us that this project did not experience problems due to the knowledge level of graduate students. "*Training done"* and "*Close monitoring"* show us the best practices followed by the researchers of the project. In this study, a second researcher is asked to code a sample of two of the interviews to check whether similar codes are

generated or not. Resulting codes were found categorically similar. This step is taken to show researcher bias was not significant on the creation of the codes.

3.4.2 Axial Coding

In the axial coding phase, we made constant comparisons among the "codes" that are formed in the open-coding phase. Some of the codes are grouped, others are aggregated or dropped, and some are not grouped, but a different type of relationships are shown among various codes. For example, codes "*Rare meetings*," "*No information flow*," "*Long distance partners*," and "*Communication issues*" can all be grouped under "*Problems with the flow of information*." By grouping different codes, we can understand the causes of such issues more efficiently. Axial coding enables us to focus on the main problems rather than list all the possible issues present in the research field. Also, the previously mentioned item, "*Sufficient experience,"* might be the result of *Training done* and *Close monitoring*. These types of codes are not aggregated but, still, a "cause and effect" type of relations among them can be shown.

3.4.3 Selective Coding

Summarization of all the finding is done in this phase of the GT and theories are formed using the axial codes, which alongside their inter-relations are used to form additional hypotheses about the problems and challenges faced in RBSPs. Alongside the issues, best practices derived from the previously completed projects are also incorporated while forming mentioned theories. However, forming these theories are not the end of our qualitative research, and the complete steps are explained below.

3.5 Steps of the Methodology

Grounded Theory does not cover the entirety of our research, and some stages come after it to validate the findings and propose a solution. All the general steps followed in this study are displayed in Figure 3.1.

These steps are:

1. The research problem is formulated after a literature review in the field.
2. From the findings that come from the literature review and the expectations of the researchers, semi-structured preliminary interviews are formed.
3. The interviews are tested on a small set of subjects to be finalized.
4. Interviews are performed with academicians that have participated in research-based software development projects at universities.
5. While conducting the interviews, data analysis is also carried out by the researchers to include open coding, axial coding, and field notes.
6. Open coding is performed, which offers a vague idea about the problems and challenges faced by the researchers and their best practices.
7. Axial coding is performed by comparing the stated problems and grouping them to form logical groups.
8. Selective coding is applied to prioritize the most critical problems, challenges, and best practices.
9. After data saturation is reached and no new coding is generated from the interviews, the interviews are terminated, and the GT methodology is used to form the final theories.
10. The results of the GT are validated by surveying the academicians of this field using a questionnaire.
11. All the previously generated data are considered, and a new methodology is proposed.
12. In addition to proposing a novel methodology, previously generated data is used to create guidelines for the PMs, team members, funding agencies and universities that will participate in RBSPs.
13. The validation of the proposed methodology is done by gathering expert opinions and including necessary additions to the method, the text, and the guidelines.

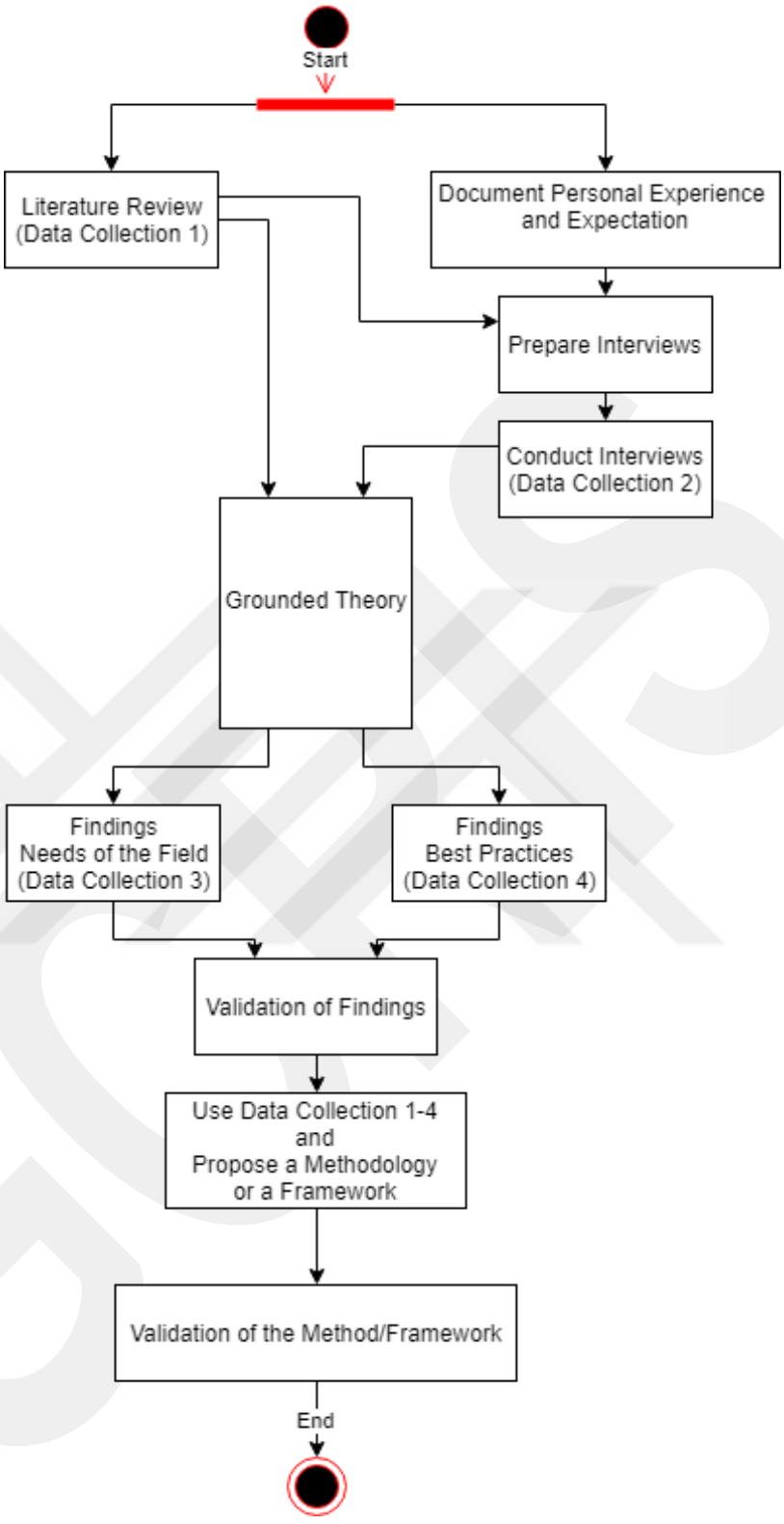


Figure 3.1 Steps of the Research Methodology

CHAPTER 4

ANALYSIS OF THE DATA

This chapter includes the analysis of the data collected via the interviews and the questionnaire. As stated earlier, previous literature on the topic shows a handful of studies that focus on research-based software development. However, these studies also clearly state the importance and the need for a new methodology for research-based projects. To this end, in the present work, semi-structured interviews are held with academicians that participated in research-based software development projects. The problems they faced during the development of the projects, challenging parts and the best practices applied by them are collected in these interviews. Also, after the analysis of the interviews, a questionnaire is distributed to validate the importance of the identified problems.

4.1 Analysis of the Interview Data

Twenty interviewees participated in fifteen different research-based software projects at universities. Ten of them were researchers, six of them graduate students and four of them were project managers in projects. The distribution of the participants' occupation within the research-based projects is shown in Figure 4.1. The interviewees are selected so that both genders are represented in the study. Seven females and thirteen males participated in the interviews with gender distribution as shown in Figure 4.2.

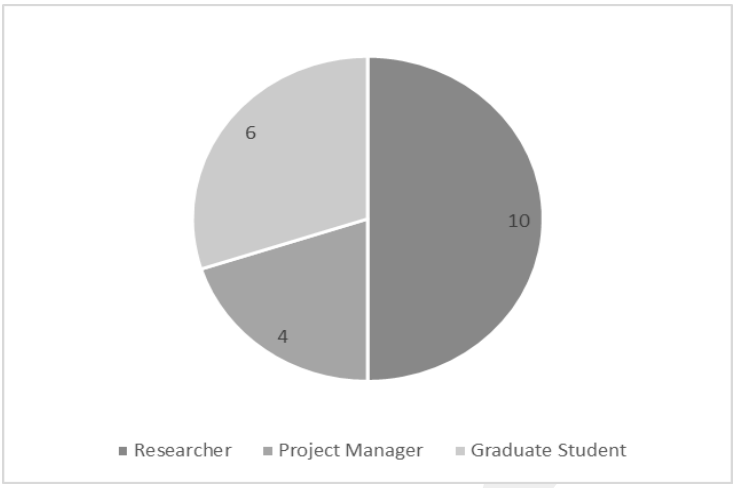


Figure 4.1 Interviewee's Role in Software Project

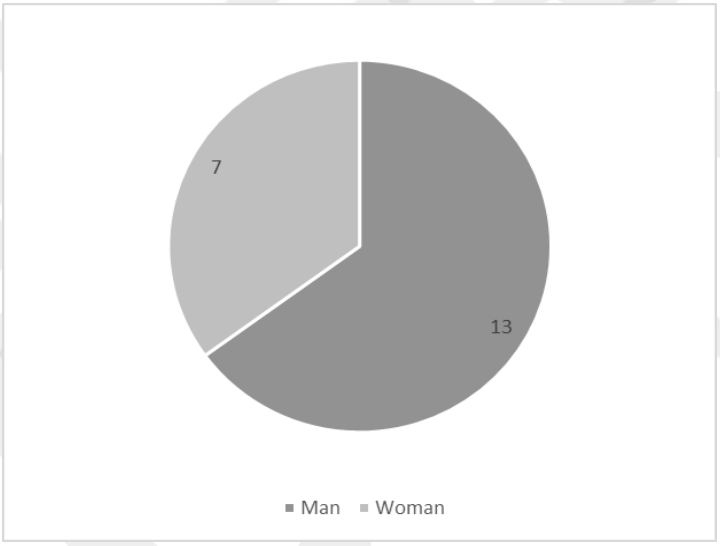


Figure 4.2 Interviewee Gender Distribution

The software projects related to the interview study were all funded: four by an international agency and the remaining eleven by a domestic one in Turkey as per the distribution in Figure 4.3. The duration of the funds varies from project to project from one and a half years to two years, two and a half years or three years as depicted in Figure 4.4.

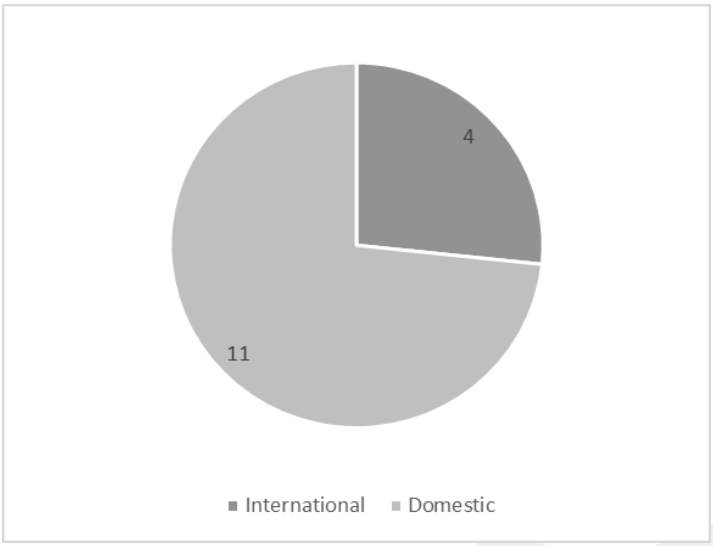


Figure 4.3 Project Funding Distribution

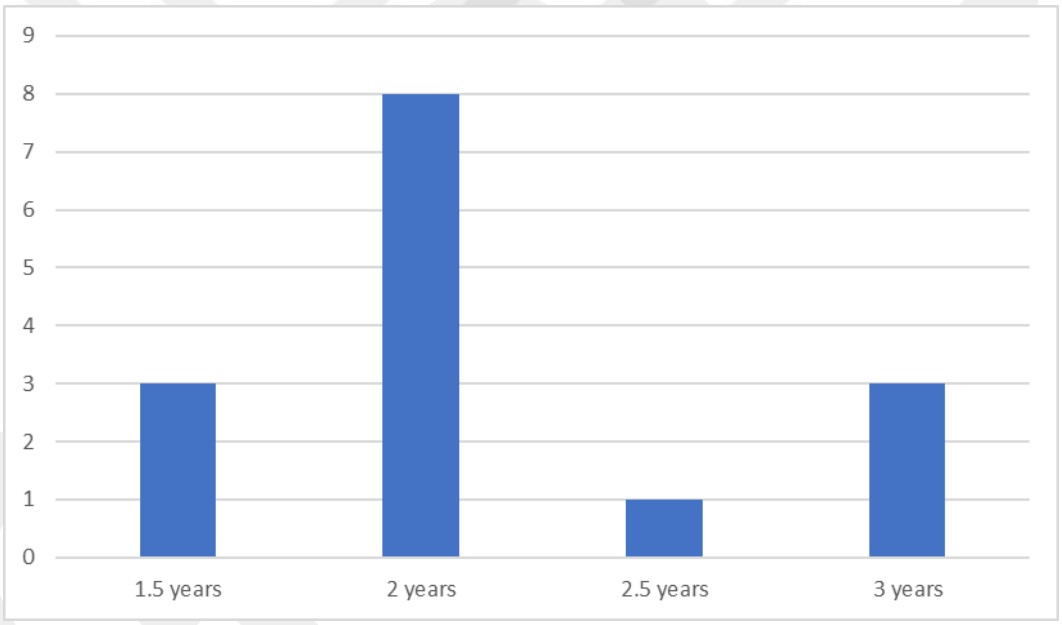


Figure 4.4 Funding Duration of the Projects

During the interviews, the participants are asked if any software development methodology is employed during the development of their RBSPs. None of the participants reported using a software development methodology in their projects. Although they claim to have not used any initially, further interviews yield that in some of the projects, development methods are used indeed. These were not common software development methods, rather those that come from the personal and previous experiences of the project managers. In four of the fifteen projects, the

personal methods employed by the researchers had similar traits to the prototyping methodology.

As stated above, the first stage of analyzing the data with GT is to code the transcribes of the interviews using Open Coding. Next, Axial Coding is done to group the codes generated in the open-coding process. The following is the list of categories that come out of the axial coding phase of GT:

1. Items Related with the Inception of the Projects
2. Problems Related with Team Members
3. Budget Problems
4. Problems Related with University
5. Communication Problems
6. Unexpected Risks
7. Project Success Factors
8. Suggestions for New Methodologies

4.1.1 Items Related with the Inception of the Projects

One can see in Figure 4.5 below the open codes *lack of detailed planning*, *insufficient initial planning*, *end-user involvement*, and *research solution/ideas* and the axial coding that is related with all of them; *Project Inception.*

In eleven of the fifteen projects, the idea of the project came directly from the project manager. The remaining four projects were initiated from the funding agencies and according to their software needs. In five of the projects, *end-user participation* was not present, and the rest of the projects had a different level of end-user involvement throughout. Most commonly, end-users are involved in the requirement gathering and testing phases. Not having end-user involvement is reported as one of the problems faced in RBSPs.

Among the fifteen projects, only three had *detailed planning* at the beginning of the project, and the remaining twelve projects had overall planning that generally includes project tasks. However, the details of the projects are decided during the development phase.

In thirteen of the fifteen projects, *possible solutions to research problems* are reported to be found in group meetings. The remaining two projects reported that the project manager decided which solutions to be pursued. Apart from the meetings and the PM, other solutions are said to come from the literature, trial and error during development, and prior experiences from similar projects.

As a result of not having detailed planning at the beginning of the project, most of the projects reported that they could follow the initial project plan by only up to 70%-80%. Project 9 (P9) is the only project that reported a dramatic *divergence from initial planning*, by following only 40% of the initial planning. Interviewee in P9 points to the requirement changes during the later stages of the project as a reason for this dramatic divergence. There is only one project that reported following the initial plan 100% which is one of the two with detailed planning at the beginning of the project.

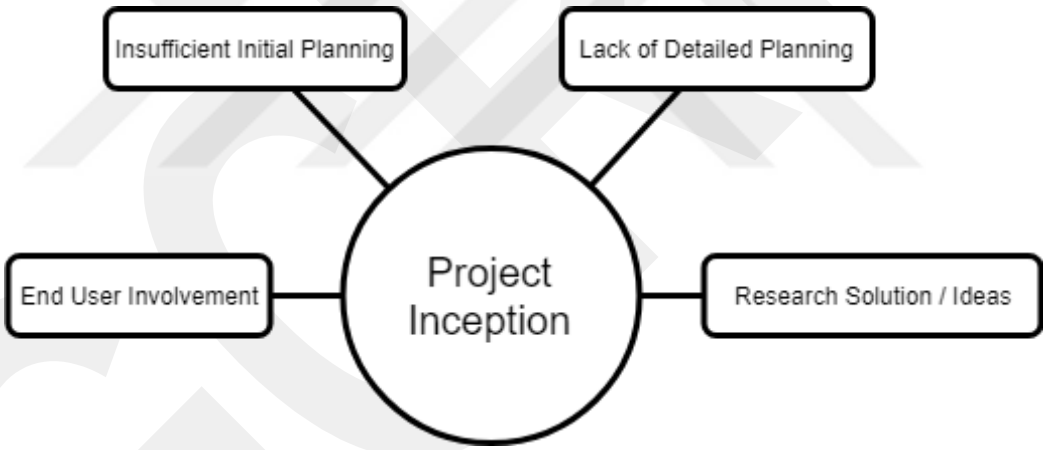


Figure 4.5 Project Inception Axial Coding

4.1.2 Problems Related with Team Members

Axial coding *Team Members* and the open codes that are related to that category are shown in Figure 4.6 below.

Some of the main problems reported in the interviews were directly related to the staff involved in these projects. *Staff turnover* in university projects is reported as one of the major problems. Especially, graduate students tend to leave the research-

based project frequently. Therefore, it is likely that if there is no reliable documentation, if the departing staff has unique skills or is responsible for a specific part of the project by themselves when they leave the project, the know-how leaves with them. P1, P3, P5, P6, and P8 reported having severe problems with the staff leaving the project. The know-how is wholly lost in certain parts and required to be re-developed in some cases, and new staff adjustment became equally problematic. In P2, P4, P7, P10, P11, P13, and P15 staff recirculation is also stated as a problem. However, the adverse effects were not as dramatic as the previously mentioned projects. The remaining four projects stated that staff recirculation was not a problem, or there was no staff leaving the project.

According to thirteen of the fifteen projects considered in the interviews, *the experience level of the graduate students is deemed low and insufficient*. Some interviewees stated this as one of the major problems of the project. In some cases, the inexperienced staff is considered in the life-cycle before the project and training of the staff is included. Unplanned training of the staff was also done in some projects, causing shifts in the schedule. There were also projects where training was not present. In such cases, graduate students learned the necessary skills from the literature, or during the development of the software.

The interviewees reported that *procuring full-time staff for the projects is hard* due to low pay. As a result of this, in the fifteen projects, only five had full-time staff, despite whose presence, still the majority comprises part-timers. Reports claim that having *part-time staff negatively affects the projects*. The part-time staff does not devote all their available time to the project and usually have other responsibilities elsewhere. University projects have relatively lower payment standards compared to the industry. This is stated to hurt the motivation of graduate students.

The interviews clearly show that the *academic hierarchy* within the project staff yields no negative effect and, to the contrary, in four of the projects, having an academic hierarchy is reported to be a positive effect on the development process.

Communication problems were one of the main problems stated in the interviews. Although mostly related to the people involved in the project, in axial-coding *communication* was stated as a separate entity to analyze it in more depth.

Figure 4.6 Problems with Team Members Axial Coding

4.1.3 Budget Problems

Axial coding *Budget Problems* and the related open codes *partial payment, insufficient funds, payment fixed for items* and *payment delays* are shown in Figure 4.7.

Seven of the fifteen projects reported *budget problems* in the interviews. Procurement problems related to funding agencies, payment dates being inconsistent, and simply not having enough money to develop the actual product are reported as the main budget problems in these projects. The other eight projects managed to use the funds effectively and did not report having any significant problems.

Commonly, the funding agencies ask for the budget plan with the application of the project to better understand the overall cost of the project. However, in reality, the items stated in the budget plan may cost less or more than the initially planned values. In such cases, project managers wish to *move the budget for specific items to other more critical items*. In the interviews, it is seen that the funding agencies are not flexible enough about switching budget between items at later stages. In RBSPs where requirements and limitations may not be known in advance, this condition proves to be a burden for PMs.

Only in three of the fifteen projects, there was a *dedicated budget for the maintenance* of the project, and the rest (twelve) are only funded for the development period by the funding agencies.

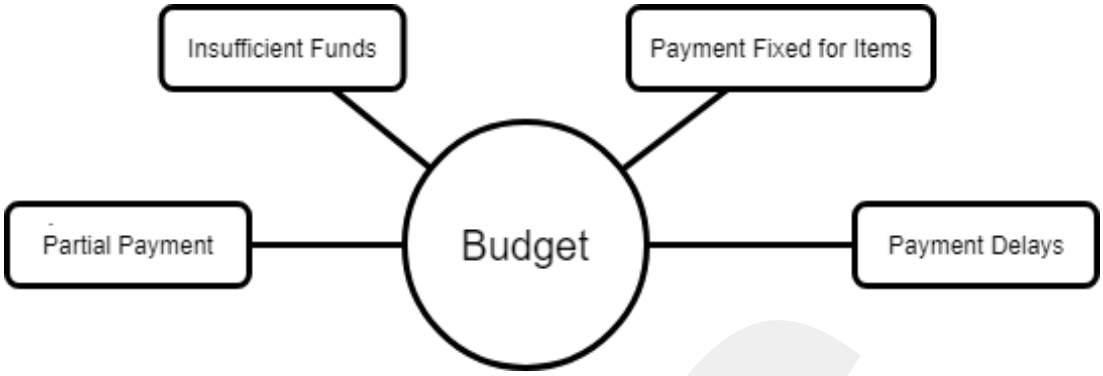


Figure 4.7 Budget Problems Axial Coding

4.1.4 Problems Related with University

The interviewees are asked about the problems faced specifically in the university projects. These items are gathered in the *University Problems* axial code below in Figure 4.8.

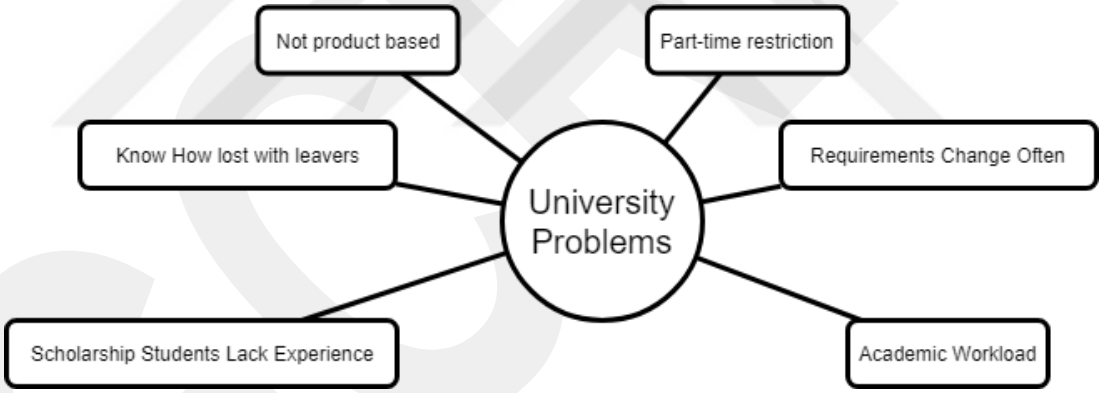


Figure 4.8 University Problems Axial Coding

The items *Part-time restriction*, *Know-how lost with leavers*, and *graduate students lack experience* were already explained as related with the *People* code. Since the interviewees listed them as the problems belonging specifically related to university-based projects, they are included under the *University Problems* as well. Most of the graduate students that participate in these projects are also research assistants at universities and attend laboratories, grade assignments and participate in proctoring multiple exams; hence, they are already loaded heavily with other tasks and utilizing them in RBSPs can be problematic for PMs.

Academic time restrictions are another hurdle, not only for graduate students but for all academicians. Since none of the project managers and researchers work as full-time on these projects, academic workload hinders their ability to work on these projects. Interviewees state that certain periods such as midterm and final weeks should be included in the project time-schedule since most of the staff will not be able to focus on the project in these weeks and so they should be handled accordingly. Any development methodology adopted for RBSPs should also consider the workload of the academicians not to overburden them if they aim to be usable.

The *Administrative works* required by the funding agencies and periodic reporting of the project are also stated as obstacles faced by academicians. During the reporting periods, the project development slows, and most of the staff are focused on preparing the required documentation. These tasks should also be included in the time-plan for the smooth operation of the project without any delays.

From the interviews, it is seen that RBSPs developed at universities are *heavily focused on the research* aspect and that the development of the product is secondary. Outputs such as papers, theses, and dissertations are the primary motivation for the staff that participates in such projects. Sometimes, schedules and priorities are shaped according to these outputs instead of finishing the product itself. This is stated as a problem unique to university settings.

In RBSPs, due to the research aspect, the *requirements can often change*. At the beginning of the project, detailed planning may not be possible since all the research details are not finalized yet. During the development of the project, research items are identified, and requirements are changed according to new findings. This is again reported as a university-specific problem by the interviewees. Any methodology that is for RBSPs should be able to cope with requirements that often change.

Low salaries, *procurement problems*, *job definition not being clear*, and *infrastructural problems* are reported as other barriers faced within university projects. As explained before, procurement problems occur because of the bureaucracy related to the funding agencies and not being able to transfer budget among payment items. Infrastructural problems occur due to the limitations of the university where the project is developed. Since these projects take up to two or three years to be completed, during this period, specific software and hardware should be

supported by the university. Interviewees reported that universities fail to support such projects from time to time and that PMs are forced to procure such items through other means.

In university projects, *job definitions may not be well-defined.* In the industry, job definitions for staff are rigid and usually focus on a single type of task at any given time. In university projects, since there is not enough budget to procure additional staff, the present members are required to participate in multiple tasks at any given time. Domain experts, requirements, testing, development, and documenting experts usually do not exist in university projects, making staff undertake most of such duties by themselves.

4.1.5 Communication Problems

Communication is stated as one of the major problems faced in RBSPs by the interviewees. The axial code *Communication* and all the related open codes are displayed in Figure 4.9 below.

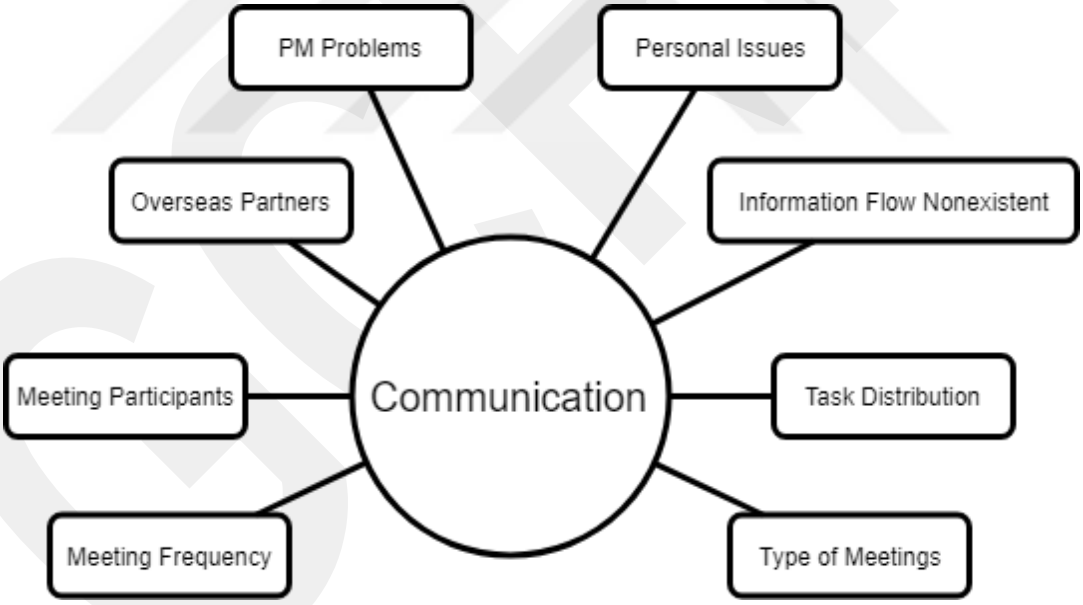


Figure 4.9 Communication Problems Axial Coding

The main communication problem reported in the interviews are the *problems with the flow of information among the members of the project*. Various reasons are given for this problem.

One of the reasons is related to the project manager. It is stated that if the *PM lacks the necessary leadership skills* and is unable to communicate with the staff effectively, the project is likely to fail. In the interviews, no such cases are pointed out, but still, the interviewees stated this scenario as a possibility.

Another likely scenario is working with *international partners* in the project. In our interviews, four of the fifteen projects were international projects, where partners from different countries participate. Communication problems happen in such cases because of the geographical limitations and the cultural differences among the development teams in different countries. Interviewees reported that communication over the Internet is not as effective as face-to-face communication, thus becoming another drawback of international projects.

Personal issues among the staff are reported as another problem related to communication. These problems are felt at the graduate student-level more frequently. The PM and the researchers are usually more dedicated to the project and tend to be more agreeable. Some graduate students are reported to only care about their theses, which is usually a part of the overall project itself. They are found to be non-cooperative with their peers. Some PMs stated that they had to speak to graduate students to solve such problems personally. In some cases, graduate students even left the project as a result.

Task distribution is reported to be one of the conventional communication problems faced in RBSPs. If the PM does not explicitly state the responsibilities of the staff according to their skillset at the beginning of the project, confusions are bound to occur, and certain parts of the projects may be neglected as a result. In P5, P7, P8, P10, P13 communications related to task distribution are reported. In P10, when the PM discovered that such a problem indeed existed in the project, they were able to quickly fix it with a meeting held specifically for the purpose. The detailed distribution of tasks to individual members is done, and the project is continued without significant drawbacks. In P7, task distribution was done at the beginning, but the skillset of the graduate students is found to be not suitable for the current distribution. A new task distribution is, therefore, done after assessing the skills of the graduate students.

The primary method of relaying information among the members of the projects is stated as the *meetings* held during the development of the projects in unison with the interviewees. The frequency of the meetings, their participants, and types of meetings are all considered to be among the critical factors for a successful project.

From the interviews, it is seen that optimal meetings should be frequent. The most successful projects reported holding weekly interviews with all members present. Although frequent meetings are regarded as overhead to the already busy schedule of the academicians, their benefits are so much that they should not be ignored. Weekly, bi-weekly, monthly, and once every six months are the most common frequencies as stated by the interviewees, who also added that meeting every six months was not enough or filled the communication gap. None of the staff had an idea about what the other members are working on and the status of the project. Meeting once every six months means the team only met with the funding agency and never without them. The interviewees that reported to have fewer problems with the flow of information said that they made participation in the meetings mandatory. In the weekly meetings, both the progress of the general project is stated, and the individual responsibilities and progress are reported. Interviewee 4 stated that having various type of meetings; such as progress meeting, task allocation meeting, problem definition meeting is also an advantage. Formatted and documented meetings can be used to track progress easily. At the beginning of the interview, a reminder of the previous meeting can be given briefly and, later, the project management task meetings and project progress meetings can be held separately with related staff. There can also be meetings where staff shows their spotting a specific research problem and possible solutions to be discussed at such gatherings.

Meeting face-to-face with all the project members may not be possible in some projects due to geographical limitations. When the participants are from different cities or different countries, face-to-face meetings cannot be done frequently with the entire staff. In such cases, interviewees stated that video conferencing over the Internet is the only alternative. Although it may not be as productive, they are still better than communicating via email or chat messages.

Any software development methodology for research-based projects is stated to have a substantial meeting schedule suitable for the staff. The frequency and type of meetings should be clarified at the beginning of the project.

4.1.6 Unexpected Risks

Some unexpected risks occur during the development of the software in research-based projects, as shown in Figure 4.10 below.

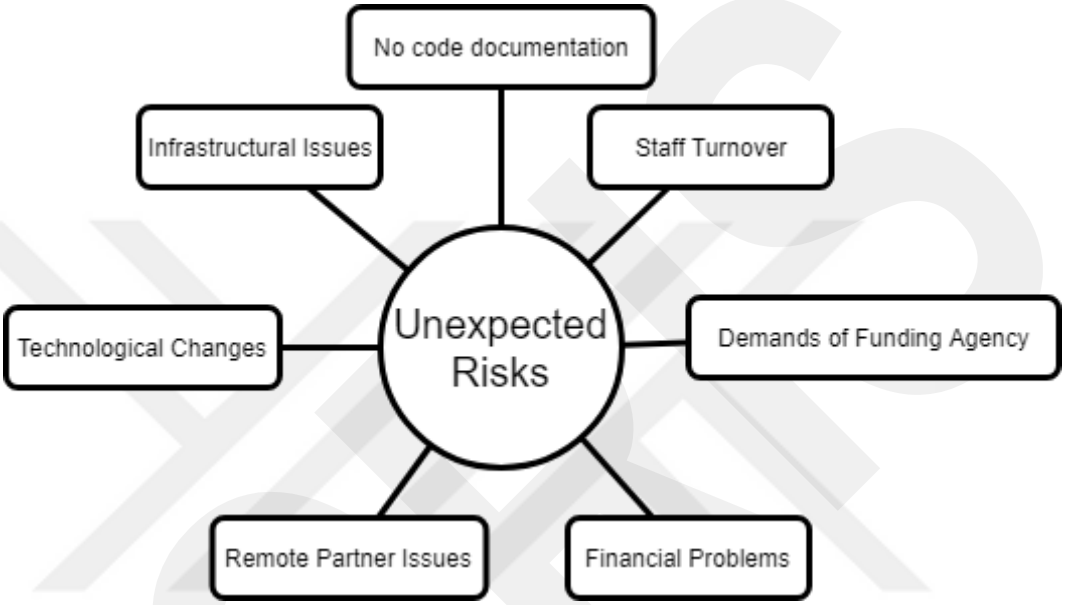


Figure 4.10 Unexpected Risks Axial Coding

Some of the unexpected risks that occur in research-based projects such as *infrastructural issues*, *staff turnover*, *financial problems*, and *remote partner issues* are mentioned in the previous items. In order to group up all the unexpected risks under a single umbrella, these items are also gathered here. Interviewees mentioned that *remote partners could be unresponsive* sometimes. In international projects where responsibility is divided among the participating countries, some parts can be incomplete or delayed due to the participants. In P6, the PM stated that this appeared to be so severe that they had to cut the funding of one of the participating universities.

Demands of the funding agencies are also reported as being a burden to the project staff. Usually, every six months progress reports are prepared and given to the

funding agency to continue receiving the funds. These reporting periods are said to have low productivity with the development of the actual project. Alongside the bureaucracy overheads, these factors are stated to be integrated into the project time-schedule in order to not be repeated later.

The participants stated that *no documentation except the ones required by the funding agencies is done*. None of the projects considered in the interviews include coding documentation. This occurs while, in an environment where staff recirculation is high, keeping documentation related to coding is a must if one intends to minimize the loss of know-how.

Technological changes are listed as another unexpected risk. In six of the fifteen projects, these changes are reported to hurt the development of the projects. Some previously found solutions became obsolete, and engine changes or new hardware requirements arose. On the other hand, technological changes are reported to have a positive effect on P7, as the inclusion of the new methods and hardware increased the efficiency of the project.

4.1.7 Project Success Factors

The *Success Factors* that are affecting RBSPs positively are also gathered from the interviewees and put in Table 4.11 below.

Figure 4.11 Success Factors Axial Coding

In fourteen of the twenty interviews, having *compatible teammates*, *a good project manager* and a *good team* are shown as factors positively affecting the success of a project. It is a common belief among the interviewees that a team that has worked together or is highly compatible will have fewer communication issues. Teams that believe in the vision of the PM will most likely embrace the project more than other instances.

The *distribution of the tasks according to the skills of the staff* is another factor reported to influence the success of research-based projects positively. In the interviews, it is seen that when task distribution is not apparent or is not suitable to the skillset of the staff, problems are likely to arise. To avoid such problems, a method to assess the skillset of the staff can be used.

All but two interviewees reported the *knowledge level of graduate students as insufficient*. To optimize the effectiveness of these individuals, constant monitoring and strict planning are suggested by the interviewees. In projects where the PM interacts with the graduate students as one-to-one or relevant researchers give mentoring and track the progress of the graduate students closely, success is deemed more likely. *Training the graduate students* both technologically and specifically for the research field resulted in a positive effect on the completion of the projects.

It is stated that the *critical parts of the projects should be identified in the early stages* of the development. These tasks should be focused on early and completed as soon as possible.

Peer review is suggested as an effective method of development since multiple individuals will have an idea of the same parts of the code and errors can be detected more efficiently [59]. On the other hand, peer reviewing the entire project is not an easy task and may quickly become cumbersome to the participants.

It has been previously mentioned that the *flow of information* relies on meetings held during project development. Holding structured meetings frequently and having a follow-up document is suggested as a factor that will positively affect the development of the research-based projects.

4.1.8 Suggestions for a New Methodology

In the interviews, the participants are asked about their suggestions to someone intending to develop a software development methodology for RBSPs. Consequently, a list of items is suggested believed to be useful for such methodologies. The items can be seen in Figure 4.12 below.

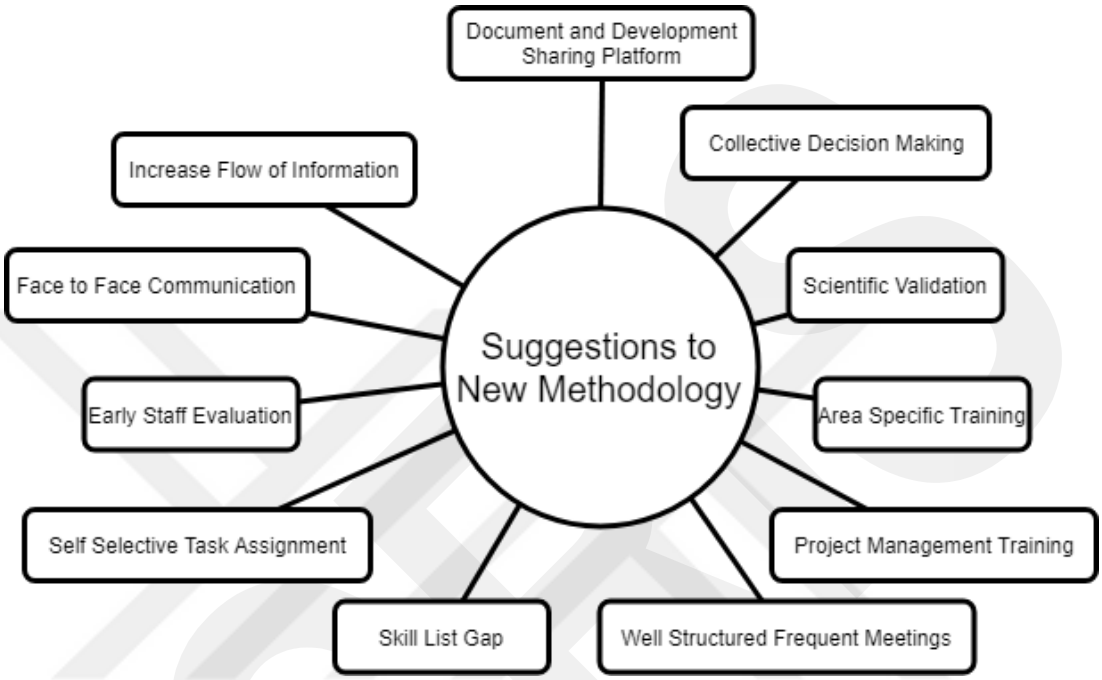


Figure 4.12 Suggestions to New Methodology

In this respect, it is suggested that the new methodology *find ways to increase the flow of information among the members of the project*. Face-to-face communication is given to increase information flow. Frequent and well-structured meetings to be held face-to-face or using video conferencing in the worst-case scenarios can also increase the flow of information.

Adapting technology was another item. *Integrating the use of document sharing, task tracking, version tracking, and online platforms* were suggested.

Another remark was that *Decision making about the research solutions should be done collectively* by the members of the project. The type of meetings should support a system, where decisions are taken and recorded so that they can be backtraced at any given time to minimize possible confusions about the decisions.

The *Skillsets of the project staff should be evaluated in the early phases* of the project so that proper task assignment can be done. A formal way of evaluating the skillset could be proposed.

A *List of skills that are necessary for the completion of the project* is suggested for comparison with that of the staff so that the gaps between the current skills and necessary skills can be found.

Coding using modern languages such as C# and Java are stated as not enough for some of the research-intensive items in the projects. *Scientific validation* of the results should be done using scientific software to compare the results with the codes generated in the project.

Training is another item suggested to be included in these projects. Both topics, specific training and project management training, are said to be useful for RBSPs. It is reported that projects that integrate area-specific training to graduate students generally perform well. Two of the projects reported that the graduate students needed to be trained about the work ethics and scheduling issues.

Despite these, not all the problems pointed out in the interviews along with the suggested solutions are directly related to software development methodologies. Still, though, all the possible problems, challenges, and best practices are collected through the interviews. The ones that are applicable will be integrated into the proposed methodology in the next chapter.

4.2 Analysis of the Questionnaire Data

The questionnaires are conducted among forty-seven individual academicians that participated in an international conference as explained before.

Among the participants, 40% stated that they have been working as an academician for 0-5 years, 26% between 6-10 years and the remaining 34% more than ten years.

Only 2% of the participants stated that they do not know about software engineering and project management; 4% stated they have little knowledge, 38% stated they have moderate knowledge, 38% stated they have good knowledge, and the remaining 17% stated they have very good knowledge about software engineering and project management.

When asked about how many research-based software development projects that the participants had been involved in as a PM, researcher or a graduate student they answered: 17% never participated, 19% only participated in a single project, 36% in two to three projects, 11% in four to five projects, and the remaining 17% in more than 5 research-based software development projects.

The questionnaire was prepared using a Likert scale with five items as 1. Not important, 2. Slightly important, 3. Moderately important, 4. Important, and 5. Very important. The participants are asked about how important they believe the listed problems are to the failure of any research-based software project.

There are six categories in the questionnaire related to the findings of the interview study; these categories are as follows:

1. Personal Information
2. Project Planning
3. Staff
4. Communication
5. Budget
6. Software Development

An analysis of the personal information category is given as the participant demographics above. The most frequent responses to the rest of the questions are given in the frequency tables per category in Tables 4.1 – 4.5.

Issues related to Project Planning are graded in Table 4.1. As can be seen from the frequency table, the participants choose "*No detailed planning done before the project*" as the major problem in the project planning phase. "*Absence of end-user or customer in the project*," "*End-user participation is low*," and "*Alternative solutions to the parts that require R&D were not set at early stages*" as the important issues. No issue is deemed as not important, slightly important or moderately important in this category.

Table 4.1 - Frequency Table of the 'Project Planning' Category

	1 - Not important	2 - Slightly important	3 - Moderately important	4 - Important	5 - Very important
1- Absence of end-user or customer in the project	2.1%	6.4%	23.4%	**53.2%**	14.9%
2- End-user participation is low in the project	0.0%	10.6%	21.3%	**46.8%**	21.3%
3- No detailed planning done prior to the project	0.0%	4.3%	17.0%	21.3%	**57.4%**
4- Alternative solutions to the parts that require R&D were not set at early stages	2.1%	4.3%	40.4%	**40.4%**	12.8%

The next category in the questionnaire is staff-related issues as per Table 4.2. Accordingly, only "*Staff recirculation is high*" is found to be a very important issue. This directly confirms the interviews conducted and the findings of previous literature [42]. The issues; "*The knowledge level of graduate students is not sufficient*", "*Most of the staff are part-time*", "*The project staff are not well-informed about project management*", "*The project staff are not well-informed in the research field*", "*The academic workloads of the project staff negatively affects the project",* and "*The skillset of the graduate students is not determined early. Accordingly task distribution is unbalanced"* are found to be important by the participants. There were no issues in this category rated as "not important," "slightly important" or "moderately important" by the majority of the participants. These findings support the idea that the problems in RBSP are usually individual-oriented.

Table 4.2 - Frequency Table of the 'Staff' Category

	1 - Not important	2 - Slightly important	3 - Moderately important	4 - Important	5 - Very important
1- The knowledge level of graduate students is not sufficient.	2.1%	6.4%	25.5%	**48.9%**	17.0%
2- Most of the staff are part-time.	4.3%	12.8%	25.5%	**51.1%**	6.4%
3- Staff recirculation is high.	0.0%	6.4%	17.0%	34.0%	**42.6%**
4- The project staff are not well-informed about project management.	0.0%	6.4%	36.2%	**42.6%**	14.9%
5- The project staff are not well-informed in the research field.	0.0%	2.1%	36.2%	**42.6%**	19.1%
6- The academic workloads of the project staff negatively affect the project.	2.1%	6.4%	31.9%	**44.7%**	14.9%
7- The skillset of the graduate students is not determined early. Accordingly task distribution is unbalanced.	0.0%	2.1%	27.7%	**46.8%**	23.4%

'*Communication*' is said to be one of the major problems in the interviews. The frequency table of the questionnaire answers to this category is given in Table 4.3 below. The issues; "*There are communication problems among project staff*" and "*The project manager does not communicate with the project team effectively*" are found to be the very important issues related to the communication. The interviews yield the flow of information between the team members as one of the most critical issues. Frequent and structured meetings are often proposed as a solution to this problem. The participants said that "*Not holding frequent, well-structured meetings regularly*" is an important problem in RBSPs. The issues; "*Communication and coordination problems with the remote partners*" and "*The project staff only have information about their parts in the project. They do not follow the progress of the project and others"* are most commonly answered as moderately important issues.

Table 4.3 - Frequency Table of the 'Communication' Category

	1 - Not important	2 - Slightly important	3 - Moderately important	4 - Important	5 - Very important
1- There are communication problems among project staff	0.0%	0.0%	12.8%	40.4%	**46.8%**
2- Not holding frequent, well-structured meetings regularly	0.0%	0.0%	19.1%	**51.1%**	29.8%
3- Communication and coordination problems with the remote partners	0.0%	8.5%	**42.6%**	31.9%	17.0%
4- The project staff only have information about their parts in the project. They do not follow the progress of the project and others	0.0%	8.5%	**44.7%**	29.8%	17.0%
5- The project manager does not communicate with the project team effectively	0.0%	2.1%	10.6%	40.4%	**46.8%**

Another set of issues are directly related to the budget problems. The answers to the issues related to the budget are given in Table 4.4 below. All the issues under the budget category; "*Insufficient budget*," "*After the initial allocation, the budget cannot be shifted between the project items*," and "*The payments are not on time, or they are partially done*" are found to be important issues to the participants.

Table 4.4 - Frequency Table of the 'Budget' Category

	1 - Not important	2 - Slightly important	3 - Moderately important	4 - Important	5 - Very important
1- Insufficient budget	0.0%	6.4%	17.0%	**42.6%**	34.0%
2- After the initial allocation, the budget cannot be shifted between the project items	2.1%	8.5%	29.8%	**40.4%**	19.1%
3- The payments are not on time or they are partially done	0.0%	8.5%	23.4%	**38.3%**	29.8%

Issues related to software development are also inquired about, and frequency table to the answers are displayed in Table 4.5 below. All five items of the software development category are found to be important issues to the participants.

Table 4.5 - Frequency Table of the 'Software Development' Category

	1 - Not important	2 - Slightly important	3 - Moderately important	4 - Important	5 - Very important
1- Not utilizing software development methodologies effectively	0.0%	0.0%	17.0%	**53.2%**	29.8%
2- Lack of documentation throughout the project	0.0%	4.3%	27.7%	**46.8%**	21.3%
3- The project management tools are not utilized properly.	0.0%	6.4%	29.8%	**46.8%**	17.0%
4- Lack of coding standards to ease the integration of newcomers into the project	0.0%	4.3%	27.7%	**42.6%**	25.5%
5- Loss of know-how with the staff leaving the project	0.0%	2.1%	14.9%	**48.9%**	34.0%

The questionnaire also included a single open-ended question asking the participants to write about any other problems that they believe are important factors for the failure of research-based projects. Few did so as per the remarks that follow:

1. Requirement analysis is not done completely.
2. There are infrastructural problems
3. There exist coordination issues
4. Participant selection is not skill-based, but relation-based.
5. Long project times. Research not being up-to-date at the end.
6. Customer requirement analysis and feasibility analysis should be done at the beginning of the project.
7. Area experts should be present in the project.

The sentiments gathered from the open-ended questions reflect the data from the interviews. Items such as “requirement analysis is not done completely” cannot be or need not be dealt with in RBSPs since the nature of R&D makes early decisions on requirements impossible [60]. This is an expected aspect of all RBSPs.

CHAPTER 5

PROPOSED METHODOLOGY

Based on the literature review and research conducted by authors through interviews and survey, we conclude that neither there is a specific software development methodology tailored to RBSPs, nor the existing methodologies are being used effectively in the present RBSPs. In [61], 'research-based project' is defined as "..... one in which the primary goal is to acquire knowledge of some kind or to resolve some kind of uncertainty, rather than producing a tangible product as a deliverable." By Research-Based Software Projects, it is meant those containing software development, at least in part, conducted in a university environment and generally managed by academicians. During the interviews conducted within this study, the interviewees who participated in such projects are asked about the key issues and challenges faced throughout their project progress. The collected results are then validated and advanced through the questionnaire by asking about the importance of these issues concerning the failure of RBSPs. Next, a novel methodology is formed - referred to as "RBAgile" - that is specifically tailored to RBSP.

In this chapter, all the features of the proposed methodology are explained, including the motivation for a new method, basic principles, details, participating actors, life-cycle, and additional guidelines for the researchers.

5.1 The motivation for a new Software Development Methodology

As stated in Chapter 2, to the best of our knowledge, the current development methodologies do not meet all the needs of the RBSPs. Furthermore, a comparison of these methodologies as provided in the section on background information also reveal that no one methodology is suitable for all types of projects and that each methodology has its weaknesses, thus preventing it from fulfilling the needs of the individuals working on RBSPs. For example, the requirements are not prone to change in Waterfall, Iterative and RAD methods; whereas the nature of RBSPs dictates often changing requirements, even in the later stages of the project. Another example could be given about agile as well as a few other traditional methods, where

those participating in the project team are expected to be highly knowledgeable and even experts in their fields. In the RBSPs, graduate students are often employed whose level of knowledge is often deemed low and inadequate. Any methodology that requires all team members to be experts may not be suitable for RBSPs since the majority of graduate students are not experts. One more example can be given about the inclusion of end-user/customer participation as mandatory, which is the case in most of the agile methods and some others, such as Prototyping and RAD. This also conflicts with the nature of RBSPs, where end-user/customer participation is not mandatory.

These incompatibilities of the current methodologies drive us to come up with a new methodology that suits the needs of individuals working on RBSPs. To do this, we first investigated the problems and best practices from the literature, and then further analyzed them by conducting interviews and questionnaire studies.

Although the literature is quite limited on the problems in RBSPs, still the author(s) managed to find two significantly relevant studies related to the factors contributing to the success of university projects [4] and the problems faced by individuals in such projects [42].

In [4], the factors that positively contribute to the success of the university projects are given in three categories, namely (i) project's initiation, (ii) development, and (iii) implementation. The most significant aspects affecting the success of the software development listed under these three categories can be given as follows:

Project initiation: problem identification, team, and interested community,

System development: team, interested community, development skills, and resources,

Implementation: team, interested community and sustainability.

In [42], people retention, lack of funds and economic uncertainty, access restriction to publication libraries, late plans on project development, getting end-user interaction, problems due to industry partnership, problems with funding agencies, and product maintenance issues raised by the user are found to be the main challenges faced by the individuals that are involved in software development within university environments.

Interviews and questionnaire results from our study showed that not only are SDM steps problematic in RBSPs but also problems usually arise related to the project management. Figure 5.1 shows the relationship between software development methodology and project management while including some other aspects. As can be seen from the figure, SDM and PM are highly integrated. Some of the development methodologies, such as Scrum, heavily concentrate on the project management aspect of the project, while others, such as Waterfall, only concentrate on the life-cycle and the steps necessary to be taken. On the contrary, in our research, we come up with both the SDM and the PM steps needed to be included in a methodology for RBSPs.

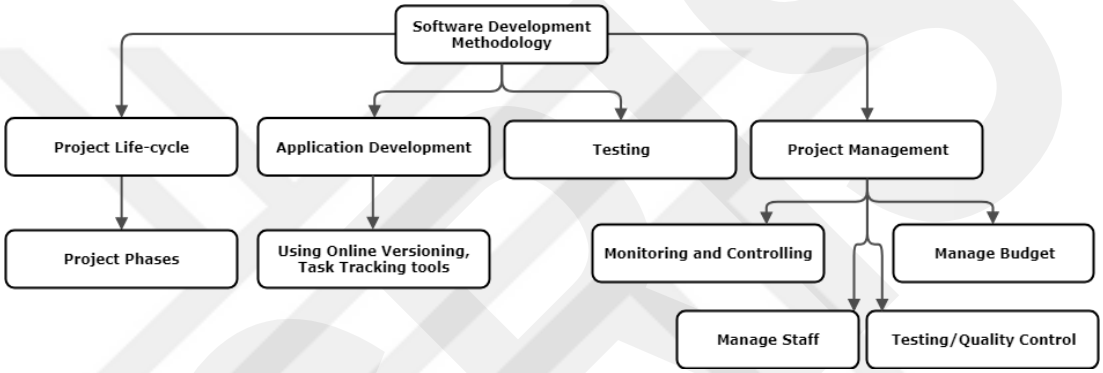


Figure 5.1 - Relation between Software Development Methodology and Project Management

The main problems as determined by the present work are listed below:

1. **Project Inception issues**: (i) End-user/customer did not exist in the project, or their participation is low; (ii) Detailed planning was not done at the beginning of the project; and (iii) Alternative solutions to the potential research problems were not set at early stages.
2. **Team Member issues** : (i) Knowledge levels of the graduate students were not sufficient; (ii) Most of the staff were part-time and, hence, did not allocate enough hours to the project; (iii) Staff turnover is high, leading to loss of know-how; (iv) Project staff were not familiar with the project management culture; (v) Academic workloads of the team members negatively affect the project; and (vi) Skillset of the graduate students are not determined early, so task distribution is problematic.

3. **Communication issues**: (i) Personal issues among team members; (ii) Coordination problems with remote partners; (iii) Project Manager lacking leadership skills; and (iv) Absence of frequent, well-structured meetings to relay information and get feedback about project issues.
4. **Budget issues**: (i) Budget was insufficient for the project; (ii) Budget could not be shifted among different procurement items after the initial distribution and (iii) Payments were either not made on time or made partially.
5. **Other issues**: (i) Software Development Methodologies were not utilized effectively; (ii) Documentation and coding standards were insufficient, and (iii) Online project management tools were not utilized properly.

Table 5.1 below shows characteristics of RBSPs, and the current SDMs are not fit for these characteristics and are not being used to solve the issues of RBSPs effectively. There are similarities between the problems gathered from the literature and those identified in this study.

Table 5.1 – Suitability of common SDMs for Research-Based Software Projects

Facts about RBSPs	Waterfall	Prototype	Spiral	RAD	Agile	
					XP	Scrum
Requirements cannot be set entirely at the beginning of RBSPs, and change often during the project phases	✕	✓	✓	✕	✓	✓
Includes inexperienced graduate students as team members	✓	✕	✓	✕	✓	✕
Absence of customer/end-user	✓	✕	✕	✕	✕	✓
Daily communication of team members is not guaranteed	✓	✓	✓	✓	✓	✕
Training of the team members is a part of the project life cycle and practices	✓	✓	✕	✓	✓	✕
Funding is generally tight and for a limited time	✕	✕	✕	✓	✓	✓
Constant monitoring, mentoring of the graduate students are required	✕	✓	✓	✕	✕	✕
Low level of operational overhead on team members.	✕	✕	✕	✕	✕	✕

Development phases are similar to general approaches used by academicians in their projects	✓	✓	×	×	×	✓
Easy to use by academicians that are NOT knowledgeable about SDMs.	×	✓	×	×	✓	✓
Should contain the solutions to the issues and challenges of RBSPs mentioned in this study	×	×	×	×	×	×

Our primary motivation is to help those working on RBSPs to carry out their projects successfully. To this end, a new methodology is proposed to focus on the problems from literature and this study and to include the best-practices collected from similar projects to fit the RBSPs needs. The issues that are not directly addressed in the proposed methodology are given as the best practices to those willing to participate in similar projects in Section 5.6 below.

5.2 Values and Principles of Research-Based Agile Software Development Methodology (RBAgile)

The proposed methodology in this chapter is called "Research-Based Agile Software Development Methodology," or RBAgile in brief. The Manifesto for RBAgile is given below and includes the values and principles of the proposed methodology.

In [62], the agile manifesto is given, where four values and twelve principles are presented and widely accepted. All the agile methods share these ideas to some degree. Some studies have modified these values and principles for their specific needs. For example, in [63] and [64] the values and the principles of the agile manifesto is modified in order to be used for scientific research purposes. In the present work, we also take four values and twelve principles of the agile manifesto

and modify this manifesto to reflect the values and principles of research-based software projects.

Below are the six values of RBAgile:

1. **Individuals and interactions** over processes and tools
2. **Working software** over comprehensive documentation
3. **Customer collaboration** over contract negotiation
4. **Responding to change** over following a plan
5. **Easy-to-use and lightweight methods** over intricate and elaborate ones
6. **Skill-oriented task allocation** over self-organizing teams.

The first four items are directly taken from the agile manifesto without any change. The first, second and fourth items are directly related with the RBSPs and should be included for any method intended to solve RBSPs' problems. The third item states the importance of customer collaboration over contract negotiation, which is also essential for RBSPs. However, the end-user or the customer may not be present in all of the RBSPs, or they might simply not be needed in some cases. This is one of the differences between the projects that utilize agile methods and RBSPs.

The fifth item is added according to the needs of the RBSPs. As stated before, the current methodologies are not efficiently used by the RBSPs. Henceforth, proposing a new method that will be complex and detailed would result in it being ignored by the individuals also involved in RBSPs. In order to be usable by RBSPs, a method should be lightweight, meaning that adopting it should not consume much time, its operation overhead should be low, and it should not entail intricate stages such that even those outside the software field can find the method easy to use.

The last item is added in the list of values to defend 'skill-oriented task allocation' referred to as one of the significant problems in RBSPs according to our interviews. Since RBSPs have limited resources, it is vital to acquire the staff that has the most suitable skills for the project and to assigns them to the tasks they can excel according to their skillset. Also, early skill assessment of the graduate students reveals the necessary areas for additional training.

The twelve principles of RBAgile can be seen on the right side of Table 5.2 below, in comparison with the twelve principles of agile.

Table 5.2 – Comparison of the Twelve Principles of Agile and the Twelve Principles of RBAgile

No	The Twelve Principles of Agile	The Twelve Principles of RBAgile	Status
1	Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.	Prioritize the satisfaction of the end-user, the customer or the project manager through early and continuous delivery of valuable software.	Modified
2	Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.	Welcome changing requirements, even late in development. Agile processes harness change which is in the nature of the Research & Development heavy software projects.	Modified
3	Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.	Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.	Same
4	Business people and developers must work together daily throughout the project.	When possible, business people, area experts, end-users, customers and team members must work together intimately throughout the project.	Modified
5	Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.	Distribute the tasks based on the skills of the individuals. Give them the environment, the support, and the training they need and monitor their progress while they carry out the job.	Replaced
6	The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.	The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.	Same

7	Working software is the primary measure of progress.	Working software is the primary measure of progress.	Same
8	Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.	Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.	Same
9	Continuous attention to technical excellence and good design enhances agility.	Continuous attention to technical excellence and good design enhances agility.	Same
10	Simplicity - the art of maximizing the amount of work not done - is essential.	Simplicity is essential, not only as the art of maximizing the amount of work not done but also employing an easy-to-understand and easy-to-use “lightweight” method.	Modified
11	The best architectures, requirements, and designs emerge from self-organizing teams.	By establishing the critical parts of the project in the early stages, and collectively coming up with the solutions to the problems emerged, a project becomes prosperous.	Replaced
12	At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.	At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.	Same

As seen from Table 5.2, principles 1, 2, 4, and 10 are modified to meet the needs of the research-based software projects. The principles numbered as 5 and 11 are our new additions to the principles of agile. The remaining six principles are kept as they are.

This modified version of the agile manifesto is a draft version called the “Manifesto for Research-Based Agile Software Development.” The latest version of this manifesto can be found online in [65].

5.3 Life-Cycle

The RBAgile Life-Cycle diagrams can be seen below in Figures 5.2 and 5.3. The diagram in Figure 5.2 shows the essential life-cycle elements of the RBAgile, whereas 5.3 displays a more detailed view of the life-cycle phases.

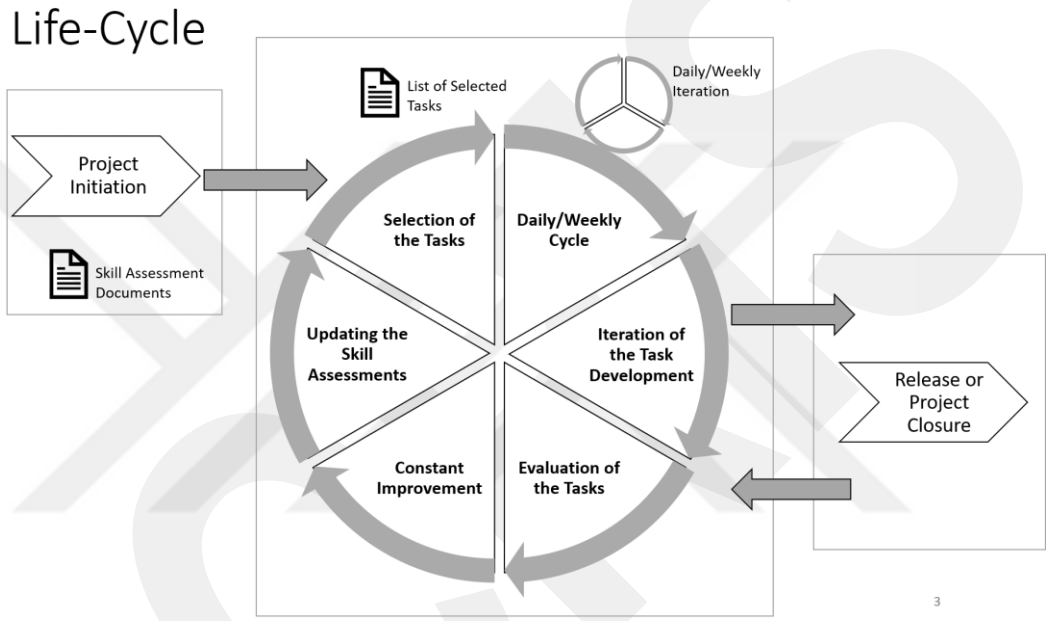


Figure 5.2 - Basic Life-Cycle Diagram of RBAgile

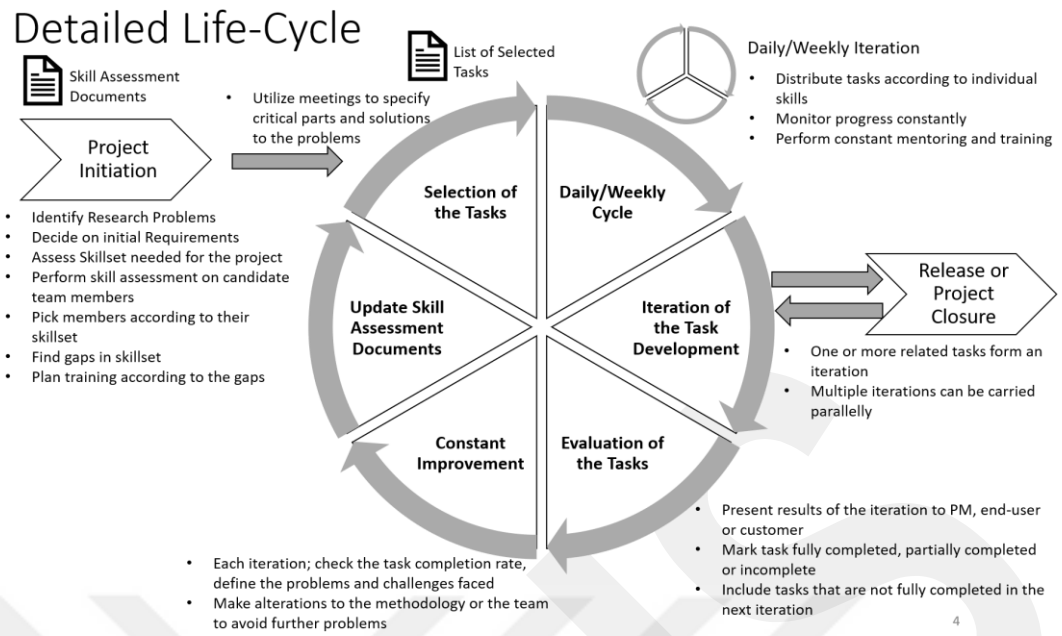


Figure 5.3 - Detailed View of the RBAgile Life-Cycle

The details of the life-cycle steps of RBAgile appear in Section 5.5. Before a detailed explanation of each step, the participating actors in RBSPs and their roles and responsibilities are defined in Section 5.4. The high quality, full page versions of the life-cycle diagrams can also be found in Appendix F and Appendix G.

5.4 Actors, Roles, and Responsibilities

Many different actors take part in the development of RBSPs some of whom, such as the project manager, researcher, and graduate students, are those that develop the project. Others can be partners, funders, authority, or the actual users of the project outputs. Not all these actors always coexist in all RBSPs. For example, the end-user or customer may not exist in all RBSP or finding an area expert to externally guide the project team may not be possible for a particular project.

5.4.1 Project Manager

The project manager (PM) is the leader of the project team [66] and always an academician. The responsibilities of the project manager predate other team members and include the initiation of the project, planning, procurement, designing, execution of the entire project, overseeing the progress and finalizing the project to mention a few [67]. In RBSPs, the idea of the project usually comes from the PM, thus making her/him the primary initiator of the project. Initial planning, finding suitable team

members, and applying for funding are common responsibilities for the PM before the project begins. During the project, the PM coordinates the team members, sets out the rules to be followed, makes task distributions, organizes meetings, communicates with the outside actors such as funding agencies and end-users, and designates some of these responsibilities to other team members. The PM is not an optional role, and all RBSPs should have one.

5.4.2 Researchers

In RBSPs researchers are academicians that have completed their doctoral studies. Some of the commonly expected responsibilities of the researchers are conducting research, finding solutions to research problems, monitoring the progress of the graduate students, educating and directing graduate students, attending the project meetings, and carrying out the duties assigned to them by the PM. Although researcher participation is not mandatory for RBSPs, most of them incorporate multiple researchers.

5.4.3 Graduate Students

These are the other members that develop RBSPs. Graduate students, as the name suggests, study masters or doctoral degrees at universities and join the project team with the aim to do research, design and develop programs, perform testing, and carry out the other responsibilities that are assigned to them by the PM and the researchers. In RBSPs, the PM can directly communicate with the graduate students in order to assign tasks and monitor their progress, or a researcher can supervise one or more graduate students and handle their participation in the project. Graduate students are generally research or teaching assistants at the same time at universities and have responsibilities there as well. A common reason for graduate students to participate in RBSPs is to research their thesis studies. In that case, the PM or one of the researchers is the advisor for the graduate student, and part of the entire RBSP is the thesis study for the graduate student. The graduate student role is not mandatory for RBSPs, but most projects involve graduate students at any rate.

5.4.4 End-User

The end-user is a person or a community that will use the output of the project, meaning that the software program is primarily designed for them [68]. End-user participation may not be possible or required due to the nature of the research being carried out in the RBSP. Whenever due, presenting requirements, updating or removing old requirements, and offering feedback to the progress of the project are the primary responsibilities that are expected from end-users. When such participation is not possible but needed, the PM should act as the end-user and take over their responsibilities.

5.4.5 Customer (Product Owner)

The customer is a role that usually could not be found in RBSPs. Due to the nature of the projects in RBSPs, they are initiated by the PM. However, in some cases, governmental institutions or industry firms may request specific R&D software to be developed. In such cases, the party that requested the R&D software is often regarded as the customer. The typical responsibilities of the customer include but are not limited to, stating their vision for the software product clearly, participating in the gathering and fine-tuning of the requirements, and participating in test phases to give feedback about the outputs [69].

5.4.6 Area Experts (Subject Matter Experts, Domain Experts)

These are the people that have authority in the area related to the research topic and possess expert-level knowledge on a discipline, technology, product, process or an entire field [70]. The primary responsibility of the area experts is to serve as a consultant in projects and share their expertise on the topics wherever required. Finding an available area expert may not be possible for all research areas, or one of the team members may already be an area expert. In these cases, external area experts do not participate in the RBSPs.

5.4.7 Funding Agencies (Project Sponsor)

Funding agencies are the institutions that sponsor the projects. Sponsoring includes supplying financial resources and providing direction. Most RBSPs are funded by such entities, whose responsibilities can include supplying funds, participating in

meetings with the project team, tracking the progress of the project, clarifying the scope of the project, and making executive decisions [71].

5.4.8 University

As mentioned before, RBSPs are carried out in a university environment. These universities also participate in RBSPs in some ways. Technology transfer offices guide academicians and help them apply to the funding agencies. In some cases, laboratories or necessary hardware are provided for projects. Universities may also have internal funding organizations that support RBSPs financially.

5.4.9 Industry Partners

Along with being a customer to an RBSP, industry firms can also be a partner to a project. When team members come together with industry partners, the task is divided among them. Industry partner responsibilities include gathering or supplying requirements, designing and developing project software, participating in meetings, providing area expertise, and other tasks. Such a partnership is not a necessary role for RBSPs. In some projects, it may be impossible to find an industry firm, or it may simply be unnecessary due to the nature of the project.

5.4.10 Partner Institutions

Both in international projects and those carried by multiple institutions domestically, managing institutions partner up with other institutions. Partner institutions are considered as team members and share the same responsibilities as the managing institution's team members. They may employ researchers and graduate students, but there is only one PM in the project based in the managing institution. As such, partner institutions do not employ PMs and can be managed by a researcher who directly communicates with the PM in the managing institution.

5.5 Detailed Explanation of Life-Cycle Steps

The steps of the RBAgile Life-cycle are:

1. Project Initiation;
2. Selection of the Tasks;
3. Daily Cycle of Development;

4. Iteration of the Task Development;
5. Evaluation of the Tasks;
6. Constant Improvement;
7. Updating the Skill Assessments; and
8. Releasing the Project Closure

Each of these steps contains multiple actions to be performed by multiple actors. The details of these steps come in what follows.

5.5.1. Project Initiation

The PM is heavily involved in the project initiation stage and identifies the research problems so as to decide on the initial requirements of the project. The PM lists the necessary skills to be able to develop such an RBSP. Candidate team members are also asked to fill the skill assessment forms. As a result, at this stage the skills needed for the project and the possible team members' skill are gathered for a comparison between the needed skills and the currently available ones. This comparison reveals the gaps in the necessary skills. Being aware of these gaps is crucial in order to plan appropriate training for the team members. In this respect, procurement of the team members is done according to their skills. The necessary skills for the project document, individual skill assessment documents, a list of initial requirements, and the list of research problems are the outputs of the Project Initiation phase.

There are many studies in the literature on how to assess the skills of the individuals. The techniques shown in some of these works, such as [72], [73], [74], [75], [76], and [77], can be used for skill assessment. Another easy way of assessing skills is merely listing the necessary skills for the project in one sheet and asking individuals to state their level of competence for each one. The answers may also be collected digitally for easy updating as the project progresses.

5.5.2. Selection of the Tasks

In this phase, team members decide on which tasks to be worked on in the next iteration. While deciding on the tasks, the importance of the tasks should be considered, and the tasks that are most critical to the project should be handled in the

early iterations of the project life-cycle. The critical tasks, updated requirements, and the solutions to the problems related to the tasks are all decided collectively throughout the meetings held with the project team members. At the end of each meeting, a list of selected tasks is generated and shared with all the team members. The tasks that are selected to be worked on in the iteration do not have to be related to software development only and project management tasks such as procurements, training, and documentation are also included in this set.

5.5.3. Daily/Weekly Cycle

In the daily/weekly cycle of the project, graduate students participate in the development of tasks that are distributed according to their skillset. The researchers oversee the progress of one or more graduate students, mentor them, and train them in order to ensure a smooth workflow. Meanwhile, the PM oversees the entire team members, assigns researchers on solving various research questions and re-arranges researcher/graduate student hierarchy upon necessity. In RBSPs, team members usually have additional responsibilities, such as academic undertakings, and work on these projects as part-time. Since RBAgile is a lightweight method, as part of its novelty the present work intends to reduce the overhead of employing an SDM. In this way, daily meetings and daily reporting of the progress is not mandatory and such operations are performed as frequently as the team members are comfortable with doing them.

5.5.4. Iteration of the Task Development

The daily/weekly cycle phase of the life-cycle repeats itself as long as the iteration takes which is usually two to four weeks. Some tasks are exceptions and may require multiple iterations to be completed. As explained before, the tasks included in the iteration phase do not have to be software development tasks and producing research items such as papers and dissertations or writing documentation for the funding agency is also a part of the iteration. Although not explicitly included in the list of selected tasks, testing and quality control steps are also a part of the iteration phase.

5.5.5. Evaluation of the Tasks

After an iteration is completed, one or more tasks related to that iteration need to be evaluated. If end-users, customers, area-experts are present, they are expected to participate in such evaluation. Otherwise, the project manager or the researcher that monitors the progress of the tasks take over as the evaluator. At the end of the evaluation, the tasks are categorized as fully completed, partially completed, or incomplete/rejected. Those deemed as fully completed are expected to be worked on in the next iteration. If there are any problems faced during the task development, such problems need to be brought to constant improvement and set for the upcoming task assignment meetings, where related team members should collectively come up with the solutions to the problems.

5.5.6. Constant Improvement

After the evaluation of each iteration ends, the effectiveness of the project itself is evaluated by the team members. The problems and challenges faced are brought to constant improvement in meetings, and changes to the employed methodology or the team members are made in order to avoid further incidents. The completion of a set of tasks may also result in changes to the requirements which need to be brought to the attention of the team members in such meetings.

5.5.7. Updating the Skill Assessments

During the project works, those student members will gain new knowledge and improve their skillset [78]. These changes should be reflected in the skill assessment document. The updates may be reflected whenever needed, but it is advised to be revised at least once in every two iterations. Based on monitorization and continuous assessment, the manager may sometimes need to delete the skills that were assumed to be possessed by the student or reduce the level of skills recorded in the document.

5.5.8. Releasing the Project/Project Closure

RBSPs are usually sponsored by funding agencies for a fixed period, and they are finalized when all tasks are completed, and the end-user, customer, or PM is satisfied, or once the funding period is ended. When there is a fund for the

maintenance of the project, the project is released at a satisfactory stage, and the project development life-cycle is continued during the maintenance of the project.

5.6 Guidelines Offered for Research-Based Software Projects

In a previous study, the author proposed a framework for university projects [42]. The framework contains essential items to consider for supervisors, research groups, and regulators. In the present study, we gather essential factors and best-practices directly from the individuals that participated in RBSPs. In Tables 5.3 to 5.6, the framework appears for the guidelines formed from the best-practices and categorized according to the role of the participants.

Table 5.3 – Framework for the Guidelines offered to Project Managers

Role: **Project Manager**	Description and Examples
Collective decision making	While finding solutions to the problems, identifying the critical parts, and evaluating the tasks always include related individuals to the decision-making progress through meetings.
Well-structured frequent meetings	Decide on a fixed interval for the meetings; ensure it suits the majority of the team members, and always record the minutes of the meetings and the decisions taken therein.
Staff procurement	When deciding whom to involve in the project, always check their skill compatibility with the needed skills for the project. Being compatible with other individuals in the project is a significant advantage. If the skills of the individuals are entirely unknown, small sample projects that require similar skills to the actual project can be given to the candidates to measure their skill levels.
Early staff evaluation	Knowing the skillset of the individuals in the project lets one plan suitable training tasks and make task distribution more balanced.
Employ online tools in project management	Online tools such as versioning, task management, continuous integration, and continuous testing to make the management of the project more comfortable and reduce the need to communicate.
Leadership skills	Projects that have a manager who communicates well with the team and shows leadership skills are deemed as successful by the team members.
Familiarity with Software Development	PM is advised to employ at least one researcher, as a team member, who is familiar with SDMs to carry the project steps successfully.
Employ a document management system	In an environment where staff turnover is frequent and expected, having a reliable document management system integrated with the task management system will help to keep the know-how within the project.

Table 5.4 – Framework for the Guidelines offered to Project Team Members

Role: **Team Member**	Description and Examples
Always communicate	Do not hesitate to ask others for help. Communication is a critical component of a successful project. The best form of communication is face-to-face.
Motivation	One should participate in parts of the project that they feel motivated about. For graduate students, writing a thesis based on the project they are involved with provides extra incentive for them.
Embrace change	RBSPs include R&D; hence, the requirements cannot be set at the beginning of the project. Be ready for some changes even in the later stages of the project.
Work ethics	Work ethics become a problem among the team members in some projects. In order not to frustrate the researchers and the project manager one responds to, one should always be on time, not forget scheduled events, be respectful to their fellow team members, and be mindful of other issues.
Pay attention to received feedback	When an end-user provides feedback, a customer, project manager or even another researcher in the project should always pay attention to such feedback as the completion of the tasks depends on these remarks.
Fulfill roles	Make sure to have enough team members that will carry the necessary roles within the method.
Reuse, Borrow	Reuse any previously developed, open source or Commercial Off-The-Shelf (COTS) module in software development to reduce time in the current project.

Table 5.5 – Framework for the Guidelines offered to End-Users/Customers

Role: **End-User/Customer**	Description and Examples
Be available	Always be available for the project team. The project team will need end-user participation in some of the meetings related to the problems, progress, or evaluation. Be there for them in order to increase productivity.
Give detailed feedback	The feedback they are trying to get from end-user is crucial for the project team. Be it in the requirements gathering phase, the progress of the project or the acceptance of the completed tasks, end-user feedback will drive the team members.

Table 5.6 – Framework for the Guidelines offered to Funding Agency and Universities

Role: **Funding Agency/ University**	Description and Examples
Consistent payments	While funding a project, make sure to be consistent with the payment schedule, and do not cut expenses without sound reasoning in order to keep the spirits of the team members high.
Flexibility among payment items	In RBSPs, it is impossible to foresee the exact amount of payment required for certain items. Let the project managers shift the money among the payment items. By doing so, the funding agency would be supporting the unexpected expenses faced in the late project phases.
Do not overburden the project team	While asking for documentation and other administrative work from the project team, funding agencies should be careful not to overburden the team members. Otherwise, the efforts of the project team are likely to shift from developing the project to replying to the required administrative works.

In Tables 5.3 to 5.6, a framework is proposed for those involved in RBSPs highlighting individuals' roles. The first column in each table shows the given best-practices for the individuals, and the second column explains the said best-practices.

5.7 Discussion of the Proposed Methodology

When proposing a new methodology for RBSPs, we decided that it should be an agile one. Previously, it was stated that no current agile methods are suitable for the needs of the RBSPs, but a considerable portion of the values and principles of agile manifesto are in line with the needs of RBSPs. What we did in the end, is to start with the agile manifesto, and modify it so that all its values and principles will apply to the RBSPs. Utilizing this modified manifesto, the findings of the literature and our research, we proposed a new methodology similar to Scrum in principle. However, the proposed methodology contains significant differences from the Scrum. In detail, as opposed to Scrum, RBAgile prioritizes distribution of the tasks according to the skills of the individuals, constant monitoring of the developers, and integrating custom training within the methodology. We also removed the mandatory daily progress meetings from the methodology. Instead, the designated team member will oversee the progress of the task development as frequently as they feel comfortable doing so. This was a necessary addition to reduce the operating overhead of the proposed methodology.

Additionally, Scrum requires the customer to be a part of the development team and work daily with them throughout the project. In RBAgile, having a customer in the project is desired, but not necessary. The absence of the customer can be covered by the project manager or researchers. Proposing a method that would not diverge from the ongoing habits of the academicians that participate in RBSPs was another concern for us. Research done yielded that dividing the project into small tasks, assigning these tasks to the developers, working closely with the developers and monitoring their progress, training the team members, and giving constant feedback are all familiar concepts for the academicians that participate in RBSPs. In another way, our method acts as an umbrella that contains these best-practices under formally defined lifecycle steps.

There is a family of a methodology, called ‘Crystal Methods,’ which also prioritize the skills of the team members [79] in project development. The Crystal Methods approach claims that formal methodologies are not necessary for developing successful projects and makes the process a secondary focus. RBAgile shares some

of its features with Crystal Methods but defends the use of formal methodologies and the necessity of following a formal life-cycle in RBSPs.

5.8 Expert Opinions

A case study is a good research method to evaluate various aspects of our proposed methodology [80]. However, in the present study, a case study approach is not utilized. The RBSPs generally takes two to three years to be completed which means that, in order to test the method on a single RBSP, a more extended period is required as compared to the present attempt. We did not have that much time at hand. Also, finding a PM that will accept the proposed methodology and utilize it in a funded project is another issue. In the present study, to evaluate the suggested RBAgile without applying a case study approach, we turned to collect expert opinions. This study defines experts as "academicians that are Assistant Professor or higher, having worked in Computer Science or Software Engineering departments for more than five years and participated in RBSPs before." Another accepted expert definition is one who is employed in any university department for five years and who has participated in more than five RBSPs.

To conduct the expert opinion study, Chapter 5 of the present study, which explains the RBAgile, and a Likert scale questionnaire is given to ten experts. Experts are asked to read the entire text and, later, answer the questionnaire using their expertise in the field. Eight experts did so, and the questionnaire can be found in Appendix H in Turkish and in Appendix I in English. In Table 5.7 below, the frequency of the questionnaire items is given. The responses to the questionnaire were (i) Strongly Disagree, (ii) Disagree, (iii) Undecided, (iv) Agree, and (v) Strongly Agree [81]. In Table 5.8, the options *strongly disagree* and *disagree* are combined, and the options *strongly agree* and *agree* are combined, leaving three aggregated responses: (i) Disagree, (ii) Undecided, (iii) Agree, to present the expert opinions in terms of positive, negative or neutral responses [82]. In the questionnaire, enough space is given after each question for the experts to provide reasoning for their answers as requested.

At the end of the questionnaire, an open-ended question is asked in order to collect all the suggestions, and criticisms experts have for the proposed methodology. The

feedback taken from the experts is reflected in the guidelines provided in Chapter 5, as well as the section explaining the RBAgile. Those parts deemed problematic in the said text are clarified by rewriting or clarifying the information differently. For example, the values and principles of the agile methodology and RBAgile were separately given in the original text. After the feedback, the items are put side by side in a table in order to show the differences more clearly. From the eight experts, seven were academicians for more than ten years and one for five to ten years. Five of the experts claimed to have average knowledge about software development methodologies; one claimed to have very good knowledge, one with good knowledge, and the remaining one little knowledge about SDMs. From all the experts, three had participated in more than five RBSPs, four in 2-3 RBSPs and one in just one RBSP. Two of the experts that had participated in more than five projects were from a department that is not related to computer science or software engineering. For our study, gathering expert opinions from outside of the related departments was essential to discover the understandability and usability of RBAgile for those not familiar with SDMs.

Table 5.7 – Frequency Table of the Expert Opinions

	1 – Strongly Disagree	2 - Disagree	3 - Undecided	4 - Agree	5 – Strongly Agree
1- RBAgile is an easy to learn method	0.00%	0.00%	0.00%	**62.50%**	37.50%
2- RBAgile is easy to use by academicians that are knowledgeable about SDMs.	0.00%	0.00%	0.00%	25.00%	**75.00%**
3- RBAgile is easy to use by academicians that are NOT knowledgeable about SDMs.	0.00%	0.00%	37.50%	**37.50%**	25.00%
4- RBAgile employs project development phases that are familiar to academicians.	0.00%	12.50%	12.50%	37.50%	**37.50%**
5- RBAgile is a useful SDM for RBSPs.	0.00%	0.00%	0.00%	37.50%	**62.50%**
6- RBAgile is more useful than the common SDMs (such as Waterfall, Scrum, XP, Spiral) for RBSPs.	0.00%	0.00%	40.00%	**40.00%**	20.00%
7- Employing RBAgile will increase the productivity of RBSPs.	0.00%	0.00%	25.00%	37.50%	**37.50%**
8- RBAgile includes all the necessary steps needed for software development in RBSPs.	0.00%	0.00%	37.50%	12.50%	**50.00%**
9- RBAgile does not convey extensive administrative costs to the team members.	0.00%	0.00%	25.00%	**50.00%**	25.00%
10- RBAgile contains the solutions to the issues and challenges of RBSPs as mentioned in the given text.	0.00%	0.00%	12.50%	37.50%	**50.00%**

Table 5.8 – Aggregated Frequency Table of the Expert Opinions

	Disagree	Undecided	Agree
1- RBAgile is an easy to learn method.	0.00%	0.00%	**100.00%**
2- RBAgile is easy to use by academicians that are knowledgeable about SDMs.	0.00%	0.00%	**100.00%**
3- RBAgile is easy to use by academicians that are NOT knowledgeable about SDMs.	0.00%	37.50%	**62.50%**
4- RBAgile employs project development phases that are familiar to academicians.	12.50%	12.50%	**75.00%**
5- RBAgile is a useful SDM for RBSPs.	0.00%	0.00%	**100.00%**
6- RBAgile is more useful than the common SDMs (such as Waterfall, Scrum, XP, Spiral) for RBSPs.	0.00%	40.00%	**60.00%**
7- Employing RBAgile will increase the productivity of RBSPs.	0.00%	25.00%	**75.00%**
8- RBAgile includes all the necessary steps needed for software development in RBSPs.	0.00%	37.50%	**62.50%**
9- RBAgile does not convey extensive administrative costs to the team members.	0.00%	25.00%	**75.00%**
10- RBAgile contains the solutions to the issues and challenges of RBSPs as mentioned in the given text.	0.00%	12.50%	**87.50%**

Table 5.7 and Table 5.8 clearly show that most of the statements on the left are agreed to by experts. Statements 1, 2, and 5 are agreed 100% so, meaning that experts agree that RBAgile can be learned quickly, can be used easily by academicians familiar with SDMs and that it is a useful SDM for RBSPs. There is only one statement in the questionnaire that received a 13% disagreement; the item "RBAgile employs project development phases that are familiar to academicians."

This means that only one expert disagreed who stated that the selection of the staff in RBAgile, which is skill-based, is significantly different from the methods they employ, so that phase of the project seemed completely unfamiliar. Statements 4, 7, 9, and 10 are agreed to by more than 75% of the experts. Statement 6 exhibits a comparison between common SDMs and RBAgile. Three experts explained that they lack the necessary knowledge on common SDMs to answer this question and left it blank. 60% of the remaining experts answered as "Agree," and the remaining 40% answered "Undecided." Following Statement 6, the statements "RBAgile is easy to use by academicians that are NOT knowledgeable about SDMs" and "RBAgile includes all the necessary steps needed for software development in RBSPs" received the highest "Undecided" rate with 38%. Based on the feedback, the authors decided to rewrite the parts of the text that explain these aspects of the proposed methodology to make them clearer. In the feedback, no significant changes are demanded by experts for the contents of the RBAgile itself, but the feedback taken is used to create additional guidelines for the PMs and team members. With the new changes done to the document, the second round of expert opinions is collected from the same eight experts in order to observe any positive changes. The results of the second round of expert opinions can be seen in Table 5.9 below. The table shows a significant improvement of acceptance rate in the collected answers. The feedback taken in the second round of expert opinions required no changes to the methodology.

Table 5.9 - Aggregated Frequency Table of the Second Round of Expert Opinions

	Disagree	Undecided	Agree
1- RBAgile is an easy to learn method.	0.00%	0.00%	**100.00%**
2- RBAgile is easy to use by academicians that are knowledgeable about SDMs.	0.00%	0.00%	**100.00%**
3- RBAgile is easy to use by academicians that are NOT knowledgeable about SDMs.	0.00%	12.50%	**87.50%**
4- RBAgile employs project development phases that are familiar to academicians.	12.50%	12.50%	**75.00%**
5- RBAgile is a useful SDM for RBSPs.	0.00%	0.00%	**100.00%**
6- RBAgile is more useful than the common SDMs (such as Waterfall, Scrum, XP, Spiral) for RBSPs.	0.00%	16.67%	**83.33%**
7- Employing RBAgile will increase the productivity of RBSPs.	0.00%	25.00%	**75.00%**
8- RBAgile includes all the necessary steps needed for software development in RBSPs.	0.00%	25.00%	**75.00%**
9- RBAgile does not convey extensive administrative costs to the team members.	0.00%	0.00%	**100.00%**
10- RBAgile contains the solutions to the issues and challenges of RBSPs as mentioned in the given text.	0.00%	12.50%	**87.50%**

CHAPTER 6

CONCLUSION AND FUTURE WORK

In this study, three main research questions are investigated, and answers to those questions are presented.

The first research question was:

1. At what level, the common software development methodologies are effectively implemented in research-based software projects?

To answer the first research question, a literature review was performed, and interviews were conducted with the individuals that participated in RBSPs. It is seen that, within university environments, no common software development methodology is employed by individuals, and that project managers and other team members usually make use of their previous knowledge about developing similar projects instead of utilizing an already known methodology. The answer to the research question can be found in Chapter 2 and Chapter 4. The literature review in Chapter 2 involves previous studies, such as [2] and [5], which display the lack of software development methodologies in projects initiated at universities. An analysis of the interviews for this study can be found in Chapter 4 based on which none of the participants appear to include a common software development methodology in their studies. Most refer to habits from previously worked projects as reflected in their RBSPs. A more in-depth investigation yielded that most of the habits participants described are similar to the already known methods, namely prototyping. Since none of the university projects highlight a common methodology used, the "At what level, the common software development methodologies are effectively implemented in Research-Based Software Projects?" is fully answered as "Common methodologies are not being utilized in RBSPs."

The second research question asks:

2. What are the common problems faced by the project team while working on research-based software projects?

The common issues and challenges faced by the project team during the development of RBSPs were gathered both from the literature, the interviews and the questionnaires conducted in this study. In Chapter 2, literature studies [4] and [42] refer to some problems related to the projects carried out by universities. In [42], these problems were put into eight categories, namely people retention, lack of funds and economic uncertainty, access restriction to publication libraries, late plans on project development, getting end-user interaction, problems due to industry partnership, problems with funding agencies, and product maintenance issues. Chapter 4 here is the analysis of our interview and questionnaire data, all of which set out the problems faced by the individuals working on RBSPs. In the interviews, twenty-four problems are categorized into six different groups as the inception of the projects, people, budget, the university, communication, and unexpected risks. Later in the questionnaire, these twenty-four problems are validated with a different set of academicians. In the result of the questionnaire, none of the twenty-four problems are deemed as unimportant or slightly important. Four problems are found to be very important, eighteen of the problems are found to be important, and the remaining two problems are found to be moderately important. Based on these findings, this research question is considered as fully answered in the study.

The last research question was:

3. What are the best practices that are employed in research-based software projects?

In Chapter 2, the literature review, the factors contributing to the success of the university projects are listed in [4]. In this study, these factors are gathered from the literature survey. In [42], the best practices of individuals have taken part in university projects are collected via interviews and later applied as a framework for the individuals that will participate in university projects.

In this study, the conducted interviews contain questions that collect the best practices of the individuals that participate in RBSPs as well as their issues and challenges. Also, in Chapter 4, the analysis of the interviews is given where the collected best practices are shown as success factors and suggestions for a new methodology. Some of the best practices include increasing the flow of information, face-to-face communication, employ well-structured frequent meetings, collective

decision making, compatible teammates, critical parts established and handled early, balanced task distribution, and so on. Accordingly, the last research question is fully answered in this study.

After answering all the research questions, the purpose of this study, which is "to develop and propose a new software development methodology that will satisfy the needs of individuals involved in research-based software projects at universities." is achieved upon the introduction of "RBAgile" in Chapter 5. The proposed methodology is based on the key issues, challenges, and the best practices gathered from the individuals having participated in RBSPs, as well as additional findings from the literature. The proposed method is considered an agile methodology by the authors since it follows and expands the values and principles of the agile manifesto. Chapter 5 shows the comparison between The Agile Manifesto and Manifesto for Research-Based Agile Software Development which is produced in the present study. RBAgile is defined with its life-cycle, actors, details of its phases, values, principles, and its discussion in Chapter 5. RBAgile is evaluated by the experts and overwhelmingly accepted as easy to learn, easy to use, and a useful method that solves stated problems for RBSPs.

As in all studies, the present work also has its limitations. To begin with, the research that is performed on this study is based on Grounded Theory, which is a well-known qualitative research method with its limitations. For instance, the findings of qualitative studies are not as scalable as quantitative studies for larger populations. To overcome the shortcoming of the qualitative studies, a questionnaire is conducted, and the findings of the interviews are validated. The quantitative data gathered from the interviews validated the findings of the results, adding more confidence to the generalizability of this study. Another limitation is due to the interviews. The selection of the participants was made through convenience sampling, which can be seen as a limitation. The number of the interviewees was high, and data saturation was reached in the interviews; though more interviews with others who have participated in the same projects could be held in order to triangulate the results better. The sample size of the questionnaire was forty-seven people. Although all of them were quality sources since they are academicians familiar with the research field, this number could be increased further to guarantee added generalizability.

Expert opinions are gathered from 8 academicians. Although the number may seem low, the feedback acquired from these academicians were all high-quality. Still, more expert opinions could provide additional insight into the issues. Another point is that the literature considered in the review section is all English with the primary sources of surveying performed on IEEE Xplore, ACM digital library, Scopus, and Google Scholar. Therefore, any literature that is written in a different language or does not show up as a result in these searches may have been overlooked. Finally, time was another limitation of this study. Since thesis studies are concluded in a limited amount of time, more thorough research could have been best. As an example, the proposed methodology could have been employed in multiple RBSPs as case-studies due to the time they take to be completed, that is two to three years on average.

In this study, three main outputs are produced: the problems faced in RBSPs, the best-practices of RBSPs, and the proposed methodology. The findings related to the problems and the best practices are believed to be useful for those who wish to participate in such projects. To know what type of challenges they may face, and what type of solutions are proposed to these challenges is considered as valuable information. These outputs may also be studied by the funding agencies as well as universities to understand the common issues and best practices better so that they can evaluate their positions in such projects and use their resources more effectively. The last output of the study is the proposed methodology. By employing RBAgile, team members are expected to increase the productivity of their development, increase the flow of information, set tasks according to the skills of the team members, and overall have a better experience developing RBSPs. Other than team members, funding agencies and universities may find the methodology useful and study the steps suggested in this method to optimize their project phases.

As a future study, the proposed methodology RBAgile can be applied to one or more RBSPs as case-studies, and the success of these projects may be monitored to discover the usefulness of the method. The proposed methodology can also be accepted as an initial draft with additions or alterations to the methodology done according to the feedback gathered from the case studies. Another future attempt may be to introduce the proposed methodology to funding agencies to modify their

methods and come up with a more suitable funding procedure that increases the success rate of research-based software projects.

Other academicians may also take the findings of this study, as well as the methodology proposed as a base point and research and expand it to suit their own specific needs.

REFERENCES

[1] C. B. Seaman, "Qualitative methods in empirical studies of software engineering," *IEEE Trans. Softw. Eng.*, vol. 25, no. 4, pp. 557–572, Jul. 1999.

[2] D. M. P. Dias, N. D. Kodikara, and M. Jayawardena, "The Need for Novel Development Methodologies for Software Projects in Universities : A Sri Lankan Case Study," *Int. J. Futur. Comput. Commun.*, vol. 2, no. 5, pp. 10–14, 2013.

[3] J. Farsi, M. Modarresi, and H. Zarea, "Obstacles and Solutions of Commercialization of University Research: Case Study of Small Businesses Development Center of University of Tehran," *J. Knowl. Manag. Econ. Inf. Technol.*, vol. 1, 2011.

[4] M. Dias, Y. Ekanayaka, and N. D. Kodikara, "Analysis of factors contributing to software development in Sri Lankan universities," *Proceedings - Pacific Asia Conference on Information Systems, PACIS 2014*. 2014.

[5] M. Dias, N. Kodikara, and Y. Ekanayaka, "Differences between universities and industry in software development," *32nd Natl. Inf. Technol. Conf. Colombo, Sri Lanka*, pp. 207–211, 2014.

[6] N. K. Denzin and Y. S. Lincoln, *Sage handbook of qualitative research*. 2000.

[7] S. J. Taylor, R. Bogdan, and M. DeVault, *Introduction to Qualitative Research Methods: A Guidebook and Resource*. Wiley, 2015.

[8] I. Deutscher, F. P. Pestello, and H. F. G. Pestello, *Sentiments and Acts*. Transaction Publishers, 1993.

[9] B. G. Glaser and A. L. Strauss, *The Discovery of Grounded Theory*. 1967.

[10] H. R. Wagner, B. G. Glaser, and A. L. Strauss, "The Discovery of Grounded Theory: Strategies for Qualitative Research.," *Soc. Forces*, 1968.

[11] M. B. Miles, A. M. Huberman, and J. Saldana, *Qualitative Data Analysis*. SAGE Publications, 2014.

[12] M. L. DESPA, "Comparative study on software development methodologies.," *Database Syst. J.*, vol. 5, no. 3, pp. 37–56, 2014.

[13] G. Elliott, *Global Business Information Technology: An Integrated Systems Approach*. Pearson Addison Wesley, 2004.

[14] Centers for Medicare & Medicaid Services, "Selecting a development

approach," *Centers Medicare Medicaid Serv.*, 2005.

[15] R. Sorensen, "A Comparison of Software Development Methodologies." p. 15, 1995.

[16] H. B. Mahapatra and B. Goswami, "Selection of Software Development Methodology (SDM): A Comparative Approach," *Int. J. Adv. Res. Comput. Sci. Softw. Eng.*, vol. 5, no. 3, pp. 58–61, 2015.

[17] M. Y. Al-Tarawneh, M. S. Abdullah, and A. B. M. Ali, "A proposed methodology for establishing software process development improvement for small software development firms," *Procedia Comput. Sci.*, vol. 3, pp. 893–897, 2011.

[18] Y. Dittrich, "What does it mean to use a method? Towards a practice theory for software engineering," *Inf. Softw. Technol.*, vol. 70, pp. 220–231, 2016.

[19] W. W. Royce, "Managing the Development of Large Software Systems: Concepts and Techniques," in *Proceedings of the 9th International Conference on Software Engineering*, 1987, pp. 328–338.

[20] J. E. Cooling and T. S. Hughes, "The Emergence of Rapid Prototyping as a Real-Time Software Development Tool," in *Second International Conference on Software Engineering for Real Time Systems*, 1989.

[21] R. Victor, "Iterative and Incremental Development :," no. June, pp. 47–56, 2003.

[22] B. W. Boehm, T. R. W. Defense, and S. Group, "A Spiral Model of Software Development and Enhancement," no. 1, 1987.

[23] Blue Ink Technical Details, "Rapid Application Development," *Republished on Developers.net October 14 2005. Rapid*, 2005. .

[24] K. Beck, "Embracing Change with Extreme Programming," *Comput. (Vol. 32 , Issue 10 , Oct 1999*, pp. 70–77, 1999.

[25] L. Rising, N. S. Janoff, and A. G. C. Systems, "The Scrum Software Development Process for Small Teams," no. August, 2000.

[26] K. Beck, *Extreme Programming Explained: Embrace Change*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2000.

[27] E. Mnkandla and B. Dwolatzky, "A survey of agile methodologies," *SAIEE Africa Research Journal*. 2004.

[28] "What is Scrum?" [Online]. Available:

https://www.scrum.org/resources/what-is-scrum. [Accessed: 22-Jan-2019].

[29] A. A. Choukou, “Agile Software Development Methodologies : A Comparative Study,” Atilim University.

[30] A. Meidan, J. A. García-García, M. J. Escalona, and I. Ramos, “Selecting Development Approach,” *Comput. Stand. Interfaces*, vol. 51, pp. 71–86, 2017.

[31] M. JAVANMARD and M. ALIAN, “Comparison between Agile and Traditional software development methodologies,” *Cumhuriyet Science Journal*. 2015.

[32] K. El Emam and G. A. Koru, “A replicated survey of IT software project failures,” *IEEE Softw.*, 2008.

[33] P. Kagan, A. Naumova, and Y. Vilman, “The Problems of project management software implementation in construction corporations,” *MATEC Web Conf.*, vol. 73, pp. 4–8, 2016.

[34] M. Niazi *et al.*, “Challenges of project management in global software development: A client-vendor analysis,” *Inf. Softw. Technol.*, vol. 80, pp. 1–19, 2016.

[35] C. Amrit and J. Van Hillegersberg, “Detecting coordination problems in collaborative software development environments,” *Inf. Syst. Manag.*, 2008.

[36] J. C. Carver and R. Prikladnicki, “Industry-Academia Collaboration in Software Engineering,” *IEEE Softw.*, vol. 35, no. 5, pp. 120–124, 2018.

[37] M. Jorgensen, “Working with Industry,” *2017 IEEE/ACM 5th Int. Work. Conduct. Empir. Stud. Ind.*, 2017.

[38] B. A. Kitchenham, T. Dyba, and M. Jorgensen, “Evidence-Based Software Engineering,” in *Proceedings of the 26th International Conference on Software Engineering*, 2004, pp. 273–281.

[39] S. E. Straus and F. A. McAlister, “Evidence-based medicine: a commentary on common criticisms,” *CMAJ*, vol. 163, no. 7, pp. 837–841, 2000.

[40] J. West and S. Gallagher, “Patterns of Open Innovation in Open Source Software,” *Open Innov. Res. a New Paradig.*, 2008.

[41] C. Joo, H. Kang, and H. Lee, “Anatomy of Open Source Software Projects :,” *5th Int. Conf. Commun. Comput. Appl.*, no. October, pp. 12–14, 2012.

[42] D. Dias, “Managing Research Based Software Product Development in Sri Lankan Universities,” no. i, 2016.

[43] N. Walliman, *Research Methods: The Basics*. Routledge, 2011.

[44] G. I. Susman and R. D. Evered, "An Assessment of the Scientific Merits of Action Research," *Adm. Sci. Q.*, vol. (23), pp. 582–603, 1978.

[45] R. Sagor, *Guiding School Improvement with Action Research*. Alexandria, Va: ASCD, 2000.

[46] J. Van Maanen, *Tales of the Field: On Writing Ethnography*. Chicago, University of Chicago Press, 1988.

[47] R. K. Yin, *Case Study Research, Design and Methods*, 3rd ed. Newbury Park: SAGE Publications, 2002.

[48] B. L. Berg, *Qualitative Research Methods for the Social Sciences*, 4th ed. S. L. Kelbaugh: Pearson Addison Wesley, 2001.

[49] J. Altmann, "Observational study of behavior: sampling methods," *Behaviour*, vol. 49, no. 3, pp. 227–67, 1974.

[50] P. I. Fusch and L. R. Ness, "Are We There Yet? Data Saturation in Qualitative Research," 2015.

[51] B. Saunders *et al.*, "Saturation in qualitative research: exploring its conceptualization and operationalization," *Qual. Quant.*, 2018.

[52] G. Guest, A. Bunce, and L. Johnson, "How Many Interviews Are Enough?: An Experiment with Data Saturation and Variability," *Field methods*, 2006.

[53] W. J. Sutherland, "Predicting the ecological consequences of environmental change: A review of the methods," in *Journal of Applied Ecology*, 2006.

[54] P. M. Kuhnert, T. G. Martin, and S. P. Griffiths, "A guide to eliciting and using expert knowledge in Bayesian ecological models," *Ecology Letters*. 2010.

[55] B. M. Ayyub, *Elicitation of expert opinions for uncertainty and risk*. 2001.

[56] E. DePoy and L. N. Gitlin, "Naturalistic Designs," *Introd. to Res.*, pp. 158–172, 2016.

[57] A. L. Strauss and J. Corbin, *Strauss, A., & Corbin, J. (1990).* 2008.

[58] R. Smyth, *Nvivo*. 2008.

[59] L. Bornmann, "Scientific peer review," *Annu. Rev. Inf. Sci. Technol.*, 2011.

[60] S. Augustine, B. Payne, F. Sencindiver, S. Woodcock, and G. Chin, *Agile Project Management: How to Succeed in the Face of Changing Project Requirements*. 2004.

[61] C. Cobb, “What is a research based project?,” 2016. [Online]. Available: https://www.quora.com/What-is-a-research-based-project. [Accessed: 21-Jan-2019].

[62] K. Beck *et al.*, “Manifesto for agile software development,” 2001.

[63] X. Amatriain, “Manifesto for Agile Research,” 2009. [Online]. Available: https://xamat.github.io/AgileResearch/. [Accessed: 19-Jan-2019].

[64] X. Amatriain, “Principles behind the Agile Research Manifesto,” 2009. [Online]. Available: https://xamat.github.io/AgileResearch/principles.html. [Accessed: 19-Jan-2019].

[65] I. Cereci, “Manifesto of RBAgile,” 2019. [Online]. Available: https://github.com/icereci/RBAgileManifesto. [Accessed: 20-Feb-2019].

[66] V. S. Anantatmula, “Project manager leadership role in improving project performance,” *EMJ - Eng. Manag. J.*, 2010.

[67] L. Crawford, “Profiling the competent project manager,” in *Proceedings of PMI Research Conference*, 2000, pp. 3–15.

[68] P. Christensson, “End User,” 2006. [Online]. Available: https://techterms.com/definition/enduser. [Accessed: 21-Jan-2019].

[69] H. S. Sverrisdottir, H. T. Ingason, and H. I. Jonasson, “The role of the product owner in scrum-comparison between theory and practices,” *Procedia-Social Behav. Sci.*, vol. 119, pp. 257–267, 2014.

[70] M.-F. Costabile, D. Fogli, C. Letondal, P. Mussio, and A. Piccinno, “Domain-expert users and their needs of software development,” in *HCI 2003 End User Development Session*, 2003.

[71] J. Bollen, D. Crandall, D. Junk, Y. Ding, and K. Börner, “From funding agencies to scientific agency,” *EMBO Rep.*, 2014.

[72] T. C. Kratschmer, P. K. Malkin, and K. Srinavas, “Method to assess the skill level of software development,” US7962890B2.

[73] D. R. Lacy, T. G. Lautzenheiser, and M. A. Bucher, “System and method for performing skill set assessment using a hierarchical minimum skill set definition,” US6524109B1.

[74] S. Varadarajan and K. K. Rao, “System and method for skill management of knowledge workers in a software industry,” US20050222899A1.

[75] L. C. e Silva and A. P. C. S. Costa, “Decision model for allocating human

resources in information system projects," *Int. J. Proj. Manag.*, 2013.

[76] S. Smee, "Skill based assessment," *BMJ*, 2003.

[77] D. L. Largent, "Measuring and Understanding Team Development by Capturing Self-assessed Enthusiasm and Skill Levels," *ACM Trans. Comput. Educ.*, vol. 16, no. 2, p. 6:1--6:27, Feb. 2016.

[78] J. H. L. Koh, S. C. Herring, and K. F. Hew, "Project-based learning and student knowledge construction during asynchronous online discussion," *Internet High. Educ.*, vol. 13, no. 4, pp. 284–291, 2010.

[79] A. Cockburn, *Crystal Clear, A Human-Powered Methodology for Small Teams*. 2004.

[80] B. Kitchenham, L. Pickard, and S. L. Pfleeger, "Case studies for method and tool evaluation," *IEEE Softw.*, vol. 12, no. 4, pp. 52–62, 1995.

[81] H. N. Boone and D. A. Boone, "Analyzing likert data," *J. Ext.*, vol. 50, no. 2, pp. 1–5, 2012.

[82] R. G. Hollingsworth, T. P. Collins, V. E. Smith, and S. C. Nelson, "Simple statistics for correlating survey responses," *J. Ext.*, vol. 49, no. 5, pp. 14–21, 2011.

APPENDIX A

SEMI-STRUCTURED INTERVIEW QUESTIONS (IN TURKISH)

Proje Hakkında Genel Sorular:

1. Projenizin konusunu kısaca özetler misiniz?
2. Projenizin için destek aldınız mı? Nereden, ne süreyle bir destek alındı?
3. Projenizde yazılım geliştirme metodolojisi olarak mevcut bir metodolojiden yararlandınız mı?
4. Destek süresi sonunda projeniz başarıya ulaştı mı?

Proje Çalışanlarıyla Alakalı Sorular:

1. Projede kaç kişi hangi posizyonlarda çalışmıştır?
2. Tam zamanlı ve yarı zamanlı çalışanlar var mıydı? Kaç tane?
3. Projeye katılanlar ne şekilde ve neden katıldılar?
4. Projenize dışardan dahil olan bir alan uzmanı var mıydı?
5. Ne sıklıkla grup toplantıları gerçekleştirebildiniz?
6. Projenizde çalışan bursiyerler ne ölçüde beklediğiniz yetkinliklere sahipti? Herhangi bir tecrübe eksikliğini gözlemlediniz mi?
    a. (EVET) Bu bir problem yarattı mı? Daha az tecrübeli kişileri projeye ne şekilde adapte edebiliriz?
    b. (HAYIR) Bu öğrencileri ne şekilde değerlendirdiniz? Proje temel kısımlarında yer aldılar mı?
7. Projede çalışanların içinde bulundukları akademik hierarşi (prof, araştırmacı, vs) proje geliştirme sürecine ne şekilde yansıdı? Nasıl bir şekilde proje yönetimini sürdürdünüz?

Araştırma Süreciyle Alakalı Sorular:

1. Yapılacak olan ARGE adımlarını detaylı olarak önceden planladınız mı?
2. (EVET)Planlama yaptıysanız; bu plana ne ölçüde uygun devam edebildiniz?
3. (EVET)Planlamaya uymadığınız yerler var ise değişikliğin nedenleri (teknoloji seçimi, araştırma yeterliliği, dış etkiler, mimari değişikliği, vb.) nelerdir?
4. Araştırma problemleri çözümünde kullandığınız fikirler nasıl ortaya çıktı?

5. ARGE gereksinimi olan bölümlerde alternatif çözümleri önceden belirlediniz mi? İlk sırada olan çözüm yeterli oldu mu? Hangi ölçüde altertatif yöntemlere başvurdunuz?

Proje Geliştirme Sürecine Dair Sorular:

1. Proje oluşum aşamasında fikir sizden mi çıktı, yoksa proje çıktısını kullanacak bir son kullanıcı projenin ilk aşamasından beri mevcut muydu?
    a. (VARDI) Proje oluşum aşamasına ne derece dahil oldular?
    b. (VARDI) Proje süresince aktif rol aldılar mı? Ne kadar süre (yüzde kaçı, kaç ay)?
    c. (HAYIR) Eksikliği hissedildi mi?
2. Başlangıçta öngördüğünüz ihtiyaçların proje süresince değişime uğradı mı? Uğradıysa; ne boyutta projeyi etkiledi?
3. (DESTEK ALINDIYSA) Destek aldığınız kurum, proje bitiminde projenin sürdürülebilmesi(maintain) için gerekli bütçe ve zaman sağladı mı? Projeyi hangi aşamada bitirmek durumunda kaldınız?
4. Bütçe açısından sorun yaşadınız mı? Yaşadıysanız bu sorun proje içeriğinizin değişmesine (kısıtlanmasına) neden oldu mu? Alternatif bütçe oluşturma gereksinimi doğdu mu?
5. Projeniz için öngördüğünüz iş-zaman planı ne ölçüde gerçekleştirildi?
6. Öngördüğünüz proje süresi herhangi bir kısıt yarattı mı?
7. Projeniz risk analizi değerlendirmesine yazdığınız unsurlar ne ölçüde beklediğiniz şekilde gerçekleşti?
8. Beklemediğiniz riskler ortaya çıktı mı? Anlatır mısınız?
9. Projenizde geliştirme süresince teknolojik değişimlere bağlı olarak etkilenme yaşandı mı?

Problemlerle Alakalı Sorular:

1. Proje süresince karşılaştığınız proje çalışanları arasında iletişime dair problemler var mıydı? Sizce bunların sebepleri nelerdir?
2. Üniversitelerde yönetilen araştırma tabanlı projeler genellikle bir araştırmaya yönelik yapılırlar ve belli bir yayın çıkarma, tez yazma gibi motivasyonla personeller bulmaktadırlar. Sizdeki çalışanların ana motivasyonu da bu şekilde miydi? Bu projeyi nasıl etkiledi?

3. Üniversite projelerinde çalışan personel genellikle kısa süreli çalışma eğilimde olup, eğitimi, araştırması sonlanınca ayrılmaktadır. Sizin proje süresince devir daim yüksek miydi? (EVET) Bunun proje geliştirme sürecine olumsuz etkileri nelerdir?
4. Sizce endüstri projelerinde karşılaşmayıp üniversite projelerinde karşılaşılan problemler nelerdir?
5. Projenin herhangi bir aşamasında karşılaştığınız başka sorunlar ve engeller nelerdir?

Yazılım Geliştirme Metodolojisi Hakkında Sorular:

1. Var olan yazılım geliştirme metotlarının araştırma tabanlı projelere ne şekilde katkısı olduğunu düşünüyorsunuz? Etkili bir şekilde kullanılabiliniyorlar mı?
2. Sizce endüstrideki ticari yazılım projeleri ile üniversitelerde yürütülen araştırma tabanlı yazılım geliştirme projeleri arasında ne gibi farklar vardır?
3. Eğer bir araştırma tabanlı yazılım geliştirme metodu geliştiriliyor olsa, geçmişteki metotlara göre ne konuda iyileştirme yapmasını önerirsiniz? Gördüğünüz ve karşılaştığınız eksikler nelerdir?

Projenizin başarılı olmasını sağlayan etkenler sizce nelerdi?

APPENDIX B

QUESTIONNAIRE (IN TURKISH)

Sayın Katılımcı,
Bu çalışma, Atılım Üniversitesi Bilgisayar Mühendisliği Bölümü Öğr. Gör. **İbrahim Cereci** tarafından yürütülmektedir. Çalışmanın amacı, üniversitelerde yürütülen **araştırma tabanlı yazılım geliştirme projelerinde karşılaşılan problemler**in tespiti ve önceliklerinin belirlenmesidir. Katılacağınız çalışmada bu tip projelerdeki tecrübelerinize veya beklentilerinize dair sorular olacaktır. Çalışmaya katılım tamamen **gönüllülük** temelindedir. Çalışmada sizden **kimlik bilgisi istenmeyecektir**. Cevaplarınız gizli tutulacak ve elde edilecek bilgiler bilimsel bilgi üretmekte kullanılacaktır. Karşılaşacağınız sorular genel olarak kişisel rahatsızlık vermeyeceği öngörülen sorulardır. Ancak herhangi bir nedenden ötürü kendinizi rahatsız hissederseniz **araştırmayı yarıda bırakmakta serbestsiniz**.

Aşağıdaki sorularda kendinize uygun olan şıkkı işaretleyiniz.

1- Ne kadar süredir akademisyen olarak çalışmaktasınız? **a) 0-5 yıl b) 6-10 yıl c) 10 yıl üzeri**
2- Yazılım Mühendisliği ve Proje Yönetimi Hakkında ne derece bilgi sahibisiniz? **a) Bilgi sahibi değilim b) Az c) Orta d) İyi e) Çok İyi**
3- Kaç adet araştırma tabanlı, yazılım geliştirme içeren projede yürütücü, araştırmacı veya bursiyer olarak yer aldınız? **a) 0 b) 1 c) 2-3 d) 4-5 e) 5+**

İlgili taraflar ile yapılan görüşmeler sonucu tespit edilen; Üniversitelerde yürütülen **Araştırma tabanlı yazılım geliştirme projeleri**nde sıklıkla karşılaşılan problemler, aşağıda kategoriler bazında listelenmektedir.
Sizden talebimiz: tecrübe ve bilgi birikiminizden hareketle, **bir projenin başarısız olmasında aşağıdaki problemlerin hangi boyutta etki edebileceğini** işaretlemenizdir.
1. Etkisiz 2. Az Etkili 3. Orta Düzey Etkili 4. Etkili 5. Çok Etkili

Proje Planlama

1- Proje fikrinin araştırmacı tarafından ortaya atılması, proje çıktısını kullanacak bir son kullanıcı bulunmaması	**1**	**2**	**3**	**4**	**5**
2- Son kullanıcının yeterince projeye dahil olmaması	**1**	**2**	**3**	**4**	**5**
3- Proje öncesinde detaylı planlama yapılmamış olması	**1**	**2**	**3**	**4**	**5**
4- ARGE gereksinimi olan bölümlerde alternatif çözümlerin	**1**	**2**	**3**	**4**	**5**

önceden yeterince belirlenmemiş olması

Personel

1- Proje bursiyerlerinin yeterli yetkinlikte olmamaları	**1**	**2**	**3**	**4**	**5**
2- Proje elemanlarının büyük çoğunluğunun yarı-zamanlı çalışıyor olması	**1**	**2**	**3**	**4**	**5**
3- Proje süresince çok sayıda ayrılan personel olması	**1**	**2**	**3**	**4**	**5**
4- Projede yer alan personelin proje yönetim süreçlerine hakim olmaması	**1**	**2**	**3**	**4**	**5**
5- Projede yer alan personelin proje araştırma alanında tecrübesiz olması	**1**	**2**	**3**	**4**	**5**
6- Proje çalışanlarının akademik yük ve sorumluluklarının projeye negatif yansımaları olması.	**1**	**2**	**3**	**4**	**5**
7- Bursiyerlerin yeteneklerinin önceden belirlenmemiş olması ve iş dağıtımının yetenekler bazında dengesiz olması	**1**	**2**	**3**	**4**	**5**

İletişim

1- Proje çalışanları arasında iletişim sorunları bulunması	**1**	**2**	**3**	**4**	**5**
2- Proje süresince yeterli sayıda nitelikli toplantı/bilgilendirme yapılmaması	**1**	**2**	**3**	**4**	**5**
3- Projedeki uzak ortakların (remote partner) iletişim ve koordinasyon problemleri	**1**	**2**	**3**	**4**	**5**
4- Çalışanların projede sadece kendi kısımları hakkında bilgi sahibi olmaları. Genel işleyişi bilmemeleri	**1**	**2**	**3**	**4**	**5**
5- Proje yöneticisinin takımla etkili şekilde iletişime geçmemesi	**1**	**2**	**3**	**4**	**5**

Bütçe

1- Bütçe yetersizliği	**1**	**2**	**3**	**4**	**5**
2- Bütçe kalemlerinin proje süresince oluşan ihtiyaçlara göre değiştirilememesi	**1**	**2**	**3**	**4**	**5**
3- Ödemelerin vaktinde ve tam yapılmaması	**1**	**2**	**3**	**4**	**5**

Yazılım Geliştirme

1- Projede yazılım geliştirme metotlarının etkili şekilde kullanılmaması.	**1**	**2**	**3**	**4**	**5**
2- Projede yeterli dökümantasyon yapılmamış olması	**1**	**2**	**3**	**4**	**5**
3- Projede, proje yönetim araçlarının düzenli kullanılmaması	**1**	**2**	**3**	**4**	**5**
4- Sonradan katılan personelin yazılıma rahatlıkla adapte olmasını sağlayacak bir kodlama standardı olmaması.	**1**	**2**	**3**	**4**	**5**
5- Projeden ayrılan yazılımcılar ile uzmanlığın (know how) gitmesi	**1**	**2**	**3**	**4**	**5**

Bu tür projelerde önemli olduğunu düşündüğünüz diğer problemleri aşağıdaki boş alana yazabilirsiniz.

APPENDIX C

CONSENT FORM (IN TURKISH)

Sayın Katılımcı,

Bu çalışma, Atılım Üniversitesi Bilgisayar Mühendisliği Bölümü Öğr. Gör. **İbrahim Cereci** tarafından yürütülmektedir. Çalışmanın amacı, **üniversitelerde yürütülen** araştırma tabanlı yazılım geliştirme projelerinin geliştirme süreçlerinde **ne tür problemlerle karşılaşıldığını**, ve bu problemlere çözüm olarak, **bu alandaki üstün yöntemleri** tespit ederek, **yeni bir yazılım geliştirme metodu önermektir**. Katılacağınız çalışmada sizlere bu tip projelerde tecrübelerinize dair sorular sorulacaktır. Bunun için uygulamacı, izninize göre size **ses kaydı** ile veya **yazılı** olarak **mülakat** yapacaktır. Bu çalışmada, sizi şaşırtmak, tutarlı-tutarsız yanıtlarınızı veya " bir açığınızı yakalamak" veya benzeri bir başka amaç yoktur. Dolayısıyla, araştırmamızın sağlıklı ve geçerli bilimsel bilgi üretebilmesi için sizden sadece, sorulara **gönül rahatlığıyla, samimi ve kendinizi en gerçekçi şekilde tanımlayacak cevaplar vermeniz beklenmektedir**.

Çalışmaya katılım tamamıyla **gönüllülük** temelindedir. Çalışmada, sizden kimlik bilgisi istenmeyecektir. Araştırmacı tarafından veri girişi ve analizi için vermiş olduğunuz bilgilerden kimliğinizi belirtecek hiçbir bilgi olmayacağı gibi, sağladığınız cevaplar sadece araştırmacı ve yardımcı araştırmacılar tarafından görülecektir. **Cevaplarınız gizli tutulacak, elde edilecek bilgiler bilimsel bilgi üretmekte kullanılacaktır**.

Çalışma boyunca karşılaşacağınız sorular, çalışmanın amacına hizmet edecek şekilde seçilmiş ve **genel olarak kişisel rahatsızlık vermeyeceği öngörülen** sorulardır. Ancak, çalışma sırasında sorulardan ya da herhangi başka bir nedenden ötürü kendinizi **rahatsız hissederseniz araştırmayı yarıda bırakmakta serbestsiniz**. Bunun için herhangi bir hak kaybına uğramayacaksınız. Böyle bir durumda uygulayıcıya araştırmadan çekilmek istediğinizi söylemeniz yeterlidir. Çalışma sonunda, araştırma ile ilgili bilgi veren bir yazılı form size okunacak, arkasından da sorularınız cevaplanacaktır.

Bu çalışmaya katıldığınız için şimdiden teşekkür ederiz. Çalışma hakkında daha fazla bilgi almak için Bilgisayar Mühendisliği Bölümü öğretim görevlisi **İbrahim Cereci** ile iletişim kurabilirsiniz (E-posta: ibrahim.cereci@atilim.edu.tr).

FORMU DOLDURUP İMZALADIKTAN SONRA TESLİM EDİNİZ.

Bu çalışmaya tamamen gönüllü olarak katılıyorum ve istediğim zaman yarıda kesip çıkabileceğimi biliyorum. Verdiğim bilgilerin bilimsel amaçlı yayınlarda kullanılmasını kabul ediyorum.

Ad, Soyad İmza Tarih
/ /2018

APPENDIX D

SEMI-STRUCTURED INTERVIEW QUESTIONS (IN ENGLISH)

General questions about the project:

1. Summarise the topic of your project?
2. Did you have funding for your project? If you did, what was the duration of the funding?
3. Did you use any software development methodology in your project? Name the methodology.
4. Did the project succeed at the end of the funding period?

Questions related to the team members:

1. How many people worked on the project? In which positions?
2. How many full-time and part-time employees were involved in the project?
3. Why did team members join the project? What was their main motivation?
4. Was there an area expert that is externally involved in the project?
5. How often did you hold meetings? What type of meetings did you hold? Who participated in the meetings?
6. Did you find the experience levels of the graduate students sufficient?
    a. (YES) Did this cause any problems? How did you utilize people with low experience levels to the project?
    b. (NO) How did you utilize these graduate students? Were they part of the core development team?
7. Did the academic hierarchy that the team members are in caused any problems?

Questions related to the research process:

1. Did you plan the R&D steps in detail before the project start date?
2. (YES) To what percentage did you follow this plan?
3. (YES) If you diverge from the plan what were the reasons? (tech choice, research incompetency, external factors, architectural changes, etc.)
4. How did you usually come up with the solutions to the research problems during development?

5. Did you prepare alternative solutions to the parts that heavily rely on R&D before the start of the project? Were your initial solutions usually applicable?

Questions related to project development phase:

1. Did the idea of the project come from the PM? Was there any end-user or customer present that were going the use the product, at the beginning of your project?
    a. (YES) How much did they involve the project?
    b. (YES) Did they actively participated in all stages of the development?
    c. (NO) Did the absence of them negatively affect the project?
2. Did the requirements you set at the beginning of the project? How did it affect the project?
3. (FUNDED) Did the funding agency give funds for the maintenance of the project? Did you have to stop development at any stage?
4. Did you face budget problems? How did these problems affect the project?
5. How well did you follow the schedule?
6. Did time limitations affect the project negatively?
7. How accurate was your initial risk assessment plan?
8. Did unexpected risks occur? Explain.
9. Did technological changes affect the project in any way?

Questions related to the problems:

1. What type of communication issued did you face between team members? What do you think were the reasons?
2. Projects that are carried by universities usually focus on research. Dissemination, thesis, and publications can be the main motivation. Was that the case for you? How did it affect the project?
3. Was the staff turnover high? How did it affect the project?
4. What are the problems that are faced at the university projects and not at the industry?
5. Explain any other problems you faced at any stages of the project.

Questions related to software development:

1. Do you think current software development methodologies are usable in a research-based software project? Are they being used effectively?
2. What kind of features would you like to see in a new software development methodology for research-based projects? What kind of shortcomings did you notice with the current methods?

What were the factors that positively affect the development of your project?

APPENDIX E

QUESTIONNAIRE (IN ENGLISH)

Dear Participant,

This study is carried by Instructor **Ibrahim Cereci** in Atilim University Computer Engineering Department. The study aims to identify and prioritize the problems that are faced during the development of research-based software projects. In this study, questions about your past experiences in similar projects or your expectations in such projects are going to be asked. Participation in the study is completely **voluntary**. No identification information will be collected in this study. Your answers will be kept confidential and will only be used to generate research data. The questions in this questionnaire are anticipated not to give disturbance to the participant. Still, if you for any reason feel uncomfortable, you are **free to quit the questionnaire at any time you wish**.

Select the item that is most suitable for you.

1- How long are you employed as an academician? **a) 0-5 years** **b) 6-10 years** **c) over 10 years**
2- How do you evaluate your knowledge level about software engineering and project management? **a) None** **b) Little** **c) Medium** **d) Well** **e) Excellent**
3- How many research-based software projects did you participate in? **a) 0** **b) 1** **c) 2-3** **d) 4-5** **e) 5+**

After holding interviews with the academicians, problems faced in research-based projects have been listed below in categories.

What we ask of you is to grade the effect of the problems listed below, according to their importance to the failure of research-based projects, according to your experience in the field.

1. Not important 2. Slightly important 3. Moderately important 4. Important 5. Very important

Project Planning

1- Lack of end-user or customer in the project	**1**	**2**	**3**	**4**	**5**
2- End-user participation is low in the project	**1**	**2**	**3**	**4**	**5**
3- No detailed planning is done before the project	**1**	**2**	**3**	**4**	**5**
4- Alternative solutions to the parts that require R&D were not set at early stages	**1**	**2**	**3**	**4**	**5**

Staff

1- Knowledge levels of graduate students are not sufficient	**1**	**2**	**3**	**4**	**5**
2- Most of the staff are part-time	**1**	**2**	**3**	**4**	**5**
3- Staff recirculation is high	**1**	**2**	**3**	**4**	**5**
4- Project staff are not well-informed about project management	**1**	**2**	**3**	**4**	**5**
5- Project staff are not well-informed in the research field	**1**	**2**	**3**	**4**	**5**
6- Academic workloads of the project staff negatively affect the project	**1**	**2**	**3**	**4**	**5**
7- Skillset of the graduate students are not determined early; task distribution is unbalanced	**1**	**2**	**3**	**4**	**5**

Communication

1- There are communication problems among project staff	**1**	**2**	**3**	**4**	**5**
2- Not holding frequent, well-structured meetings regularly	**1**	**2**	**3**	**4**	**5**
3- Communication and coordination problems with the remote partners.	**1**	**2**	**3**	**4**	**5**
4- Project staff only have information about their parts in the project. They do not follow the progress of the project and others	**1**	**2**	**3**	**4**	**5**
5- Project manager does not communicate with the project team effectively	**1**	**2**	**3**	**4**	**5**

Budget

1- Insufficient budget	**1**	**2**	**3**	**4**	**5**
2- After the initial allocation, the budget cannot be shifted between the project items	**1**	**2**	**3**	**4**	**5**
3- Payments are not on time or partially done	**1**	**2**	**3**	**4**	**5**

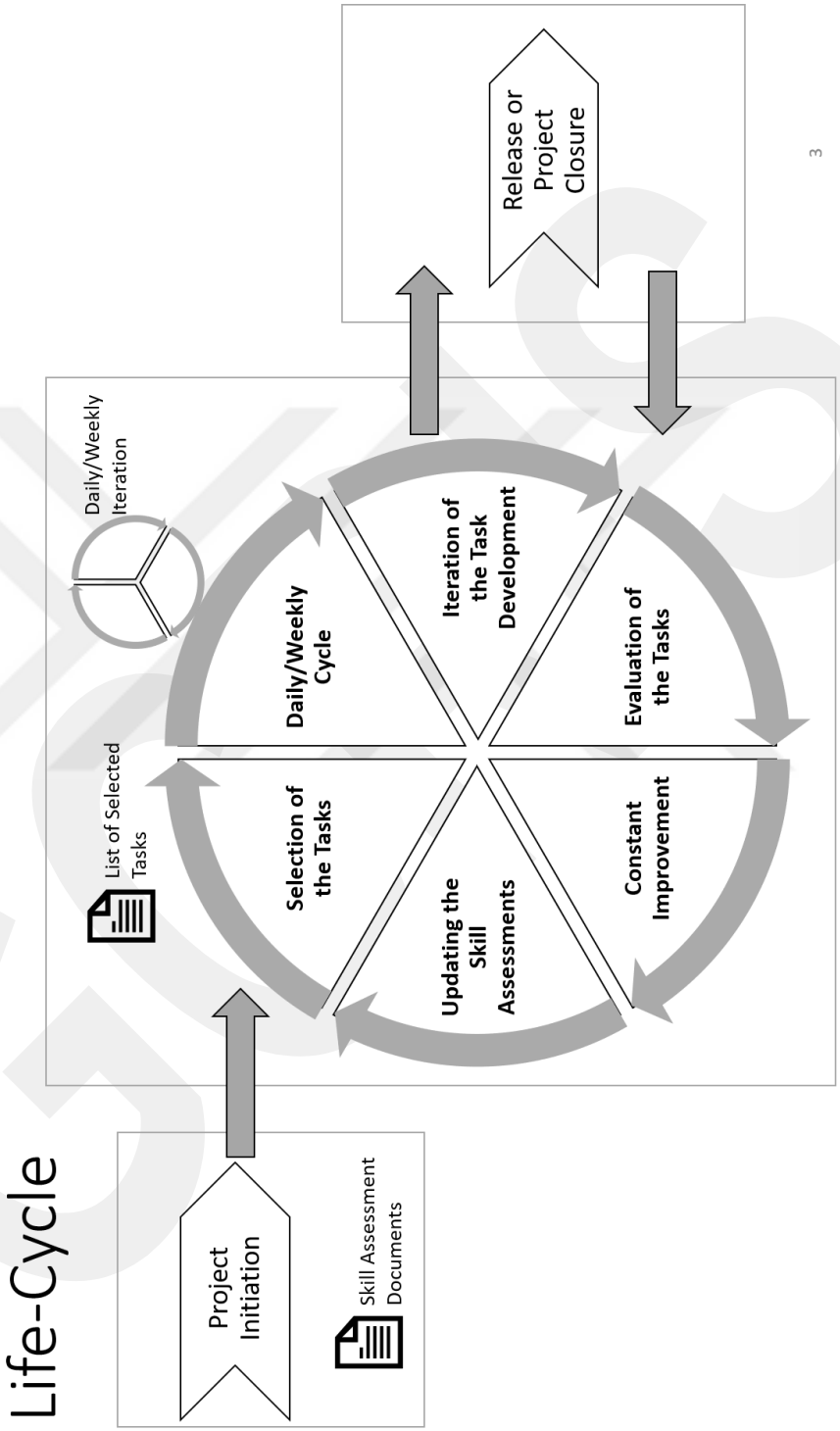
Software Development

1- Not utilizing software development methodologies effectively	**1**	**2**	**3**	**4**	**5**
2- Lack of documentation throughout the project	**1**	**2**	**3**	**4**	**5**
3- Project management tools are not utilized properly	**1**	**2**	**3**	**4**	**5**
4- Lack of coding standard that will ease the integration of newcomers to the project	**1**	**2**	**3**	**4**	**5**
5- Loss of know-how with the staff leaving the project	**1**	**2**	**3**	**4**	**5**

What are some other problems that you believe are important factors for the failure of research-based projects?

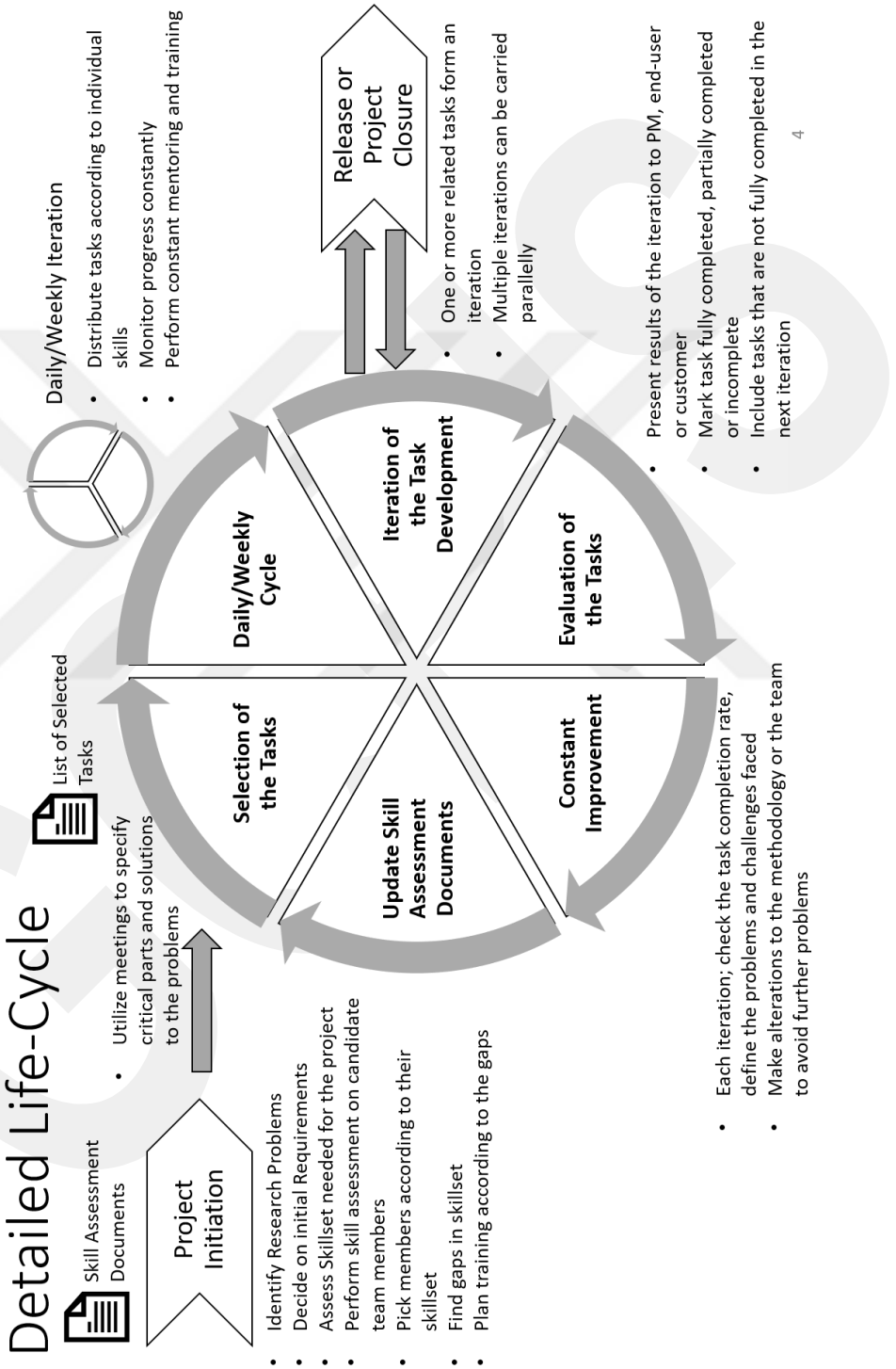
APPENDIX F

BASIC LIFE-CYCLE OF "RBAgile"



APPENDIX G

DETAILED VIEW OF THE "RBAgile" LIFE-CYCLE



APPENDIX H

QUESTIONNAIRE FOR EXPERT OPINIONS (IN TURKISH)

Sayın Katılımcı,
Bu çalışma, Atılım Üniversitesi Bilgisayar Mühendisliği Bölümü Öğr. Gör. **İbrahim Cereci** tarafından yürütülmektedir. Çalışmanın amacı, **üniversitelerde yürütülen araştırma tabanlı yazılım geliştirme içeren projelerinde kullanılmak için önerilmiş olan RBAgile** adlı yazılım geliştirme metodunu birçok farklı boyuttan değerlendirmektir. Metot ile alakalı **tarafınıza sunulan dokümanı okuduktan sonra**, bu tip projelerdeki tecrübelerinize veya beklentilerinize dair sorular olacaktır. Çalışmaya katılım tamamen gönüllülük temelindedir. Cevaplarınız gizli tutulacak ve elde edilecek bilgiler bilimsel bilgi üretmekte kullanılacaktır. Karşılaşacağınız sorular genel olarak kişisel rahatsızlık vermeyeceği öngörülen sorulardır. Ancak herhangi bir nedenden ötürü kendinizi rahatsız hissederseniz **araştırmayı yarıda bırakmakta serbestsiniz**.
Aşağıdaki sorularda kendinize uygun olan şıkkı işaretleyiniz.

1- Ne kadar süredir akademisyen olarak çalışmaktasınız? **a) 1-5 yıl b) 6-10 yıl c) 10 yıl üzeri**
2- Yazılım Mühendisliği ve Proje Yönetimi Hakkında ne derece bilgi sahibisiniz? **a) Bilgi sahibi değilim b) Az c) Orta d) İyi e) Çok İyi**
3- Kaç adet araştırma tabanlı, yazılım geliştirme içeren projede yürütücü, araştırmacı veya bursiyer olarak yer aldınız? **a) 0 b) 1 c) 2-3 d) 4-5 e) 5+**

Üniversitelerde yürütülen araştırma tabanlı yazılım geliştirme projelerinde kullanılmak üzere geliştirilmiş olan RBAgile adlı yazılım geliştirme metoduna dair aşağıda birtakım cümleler verilmektedir. Sizden talebimiz: metodun açıklandığı dokümanı okuduktan sonra tecrübe ve bilgi birikiminizden hareketle, **RBAgile hakkında aşağıda belirtilen cümlelere ne derece katıldığınızı** işaretlemenizdir. İşaretlediğiniz şıklara dair EK AÇIKLAMA gereksinimi duymanız halinde sorunun hemen altında bulunan açıklama bölümünü kullanabilirsiniz. Özellikle katılmadığınız sorularda açıklama yapmanız, çalışmamızı daha ileri noktaya götürmede önemli katkılar sağlayacaktır.
Birden beşe kadar olan seçeneklerin tam karşılıkları aşağıda belirtildiği şekildedir.
1. Kesinlikle Katılmıyorum
2. Katılmıyorum

3.Kararsızım
4. Katılıyorum
5. Kesinlikle Katılıyorum

1- RBAgile öğrenmesi kolay bir metotdur. Açıklama:	**1 2 3 4 5**
2- RBAgile, yazılım mühendisliği alanı uzmanı akademisyenler tarafından kolaylıkla kullanılabilir. Açıklama:	**1 2 3 4 5**
3- RBAgile, yazılım geliştirme metotları hakkında bilgi sahibi olmayan akademisyenler tarafından dahi kolaylıkla kullanılabilir. Açıklama:	**1 2 3 4 5**
4- RBAgile, akademisyenlerin proje geliştirmeye dair alışık oldukları yöntemlere uygundur. Açıklama:	**1 2 3 4 5**
5- RBAgile, araştırma tabanlı yazılım geliştirme projeleri için kullanışlı bir metottur. Açıklama:	**1 2 3 4 5**
6- RBAgile, araştırma tabanlı yazılım geliştirme projeleri için, yaygın yazılım geliştirme metotlarına kıyasla (örn: waterfall, SCRUM, XP, Spiral) daha kullanışlıdır. Açıklama:	**1 2 3 4 5**
7- RBAgile metodunun kullanılması, araştırma tabanlı yazılım geliştirme projelerindeki verimliliği artıracaktır. Açıklama:	**1 2 3 4 5**

8- RBAgile metodu, akademisyenlerin yazılım geliştirme projelerinde ihtiyaç duyacakları tüm yazılım geliştirme basamaklarını kapsar. Açıklama:	**1 2 3 4 5**
9- RBAgile, proje çalışanlarına metodun uygulanmasından kaynaklı ağır ek yük getirmez. Açıklama:	**1 2 3 4 5**
10- RBAgile, okuduğunuz metinde tanımlanmış olan araştırma tabanlı proje problemlerinin çözümlerini kapsar. Açıklama:	**1 2 3 4 5**

RBAgile ile alakalı her türlü eleştiri, ekleme ve değişiklik önerilerinizi lütfen aşağıda belirtiniz.

Ayrıca eksik veya yanlış olduğunu düşündüğünüz kısımları direkt tarafınıza iletilmiş olan metin üzerinden de gösterebilirsiniz.

Çalışmaya katıldığınız için teşekkür ederiz.

APPENDIX I

QUESTIONNAIRE FOR EXPERT OPINIONS (IN ENGLISH)

Dear Participant,

This study is carried by Instructor **Ibrahim Cereci** in Atilim University Computer Engineering Department. The study aims to evaluate various aspects of the RBAgile, which is the proposed research-based software development methodology by the researcher. After reading the given document that is related with RBAgile, please state your level of agreement to the below statements about RBAgile. Use your past experiences in similar projects or your expectations in such projects. Participation in the study is completely **voluntary**. No identification information will be collected in this study. Your answers will be kept confidential and will only be used to generate research data. The questions in this questionnaire are anticipated not to give disturbance to the participant. Still, if you for any reason feel uncomfortable, you are **free to quit the questionnaire at any time you wish**.

Select the item that is most suitable for you.

1- How long are you employed as an academician? **a) 1-5 years** **b) 6-10 years** **c) over 10 years**
2- How do you evaluate your knowledge level about software engineering and project management? **a) None** **b) Little** **c) Medium** **d) Well** **e) Excellent**
3- How many research-based software projects did you participate in? **a) 0** **b) 1** **c) 2-3** **d) 4-5** **e) 5+**

Please use explanation fields to justify your reasoning, especially if you answer the question as strongly disagree, disagree or undecided.

1. Strongly Disagree
2. Disagree
3.Undecided
4. Agree
5. Strongly Agree

1- RBAgile is an easy to learn methodology Explanation:	**1 2 3 4 5**
2- RBAgile can easily be used by the academicians that are knowledgeable about software engineering. Explanation:	**1 2 3 4 5**
3- RBAgile can easily be used by the academicians that are NOT knowledgeable about software engineering. Explanation:	**1 2 3 4 5**
4- RBAgile includes development and management steps that are familiar to the academicians. Explanation:	**1 2 3 4 5**
5- RBAgile is a useful method for research-based software development projects. Explanation:	**1 2 3 4 5**
6- RBAgile is more useful than the common software development methodologies (e.g., waterfall, Scrum, XP, Spiral) for research-based software projects. Explanation:	**1 2 3 4 5**
7- Using RBAgile in research-based software projects should increase the productivity of the development. Explanation:	**1 2 3 4 5**

8- RBAgile includes all the necessary steps for software development in research-based software projects. Explanation:	**1 2 3 4 5**
9- RBAgile does not bring excessive operation overhead to the project development team. Explanation:	**1 2 3 4 5**
10- RBAgile contains the solutions for all of the problems that are stated in the given text. Explanation:	**1 2 3 4 5**

Please write any criticism, request, and corrections about RBAgile below.

Thank you for attending this study.