DEVELOPING A NEW METHODOLOGY FOR SOFTWARE PROJECTS

A Thesis Submitted to the
Graduate School of Natural and Applied Sciences of Dokuz Eylul University
in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy in Computer Engineering, Computer Engineering Program

by
Kökten Ulas BIRANT

April, 2006
IZMIR

Ph.D. THESIS EXAMINATION RESULT FORM

We have read the thesis entitled **"DEVELOPING A NEW METHODOLOGY FOR SOFTWARE PROJECTS"** completed by **KÖKTEN ULAS BIRANT** under supervision of **PROF. DR. ALP KUT** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Doctor of Philosophy.

Prof. Dr. Alp KUT

Supervisor

Committee Member

Committee Member

Jury Member

Jury Member

Prof.Dr. Cahit HELVACI

Director

Graduate School of Natural and Applied Sciences

ACKNOWLEDGMENTS

I would like to express my sincere appreciation first and foremost to my advisor, Prof. Dr. Alp KUT, for his strong support, patience, valuable insights and encouragement, not only in bringing this research work to a successful completion, but also in all aspects of my academic life. He devoted his time and energy to improve this thesis despite his busy schedule. Especially in desperate time, this thesis is completed with his physical and moral support.

I extend my thanks to the members of my committee, Asst. Prof. Dr. Sen ÇAKIR, and Asst. Prof. Dr. Damla KUNTALP for their useful comments and suggestions during my study. They prepared a path for a successful Ph.D. thesis and with their valuable support; the “Rings” has been completed.

I would like to thank also to Assoc. Prof. Dr. Yalçin ÇEBI, especially for his encouraged discussions. His support on writing the thesis is also important, but I should thank to him for his moral support.

I have two more moral supporters in this work, after my family: Fenerbahçe, which I am proud to be a member of and Efes Rotary Club, which brings me valuable friendships. I should thank to all my friends in these clubs, but I will only thank to Fenerbahçe and Efes Rotary Club, because they provide my friends, who were always with me with their support, while I am exhausted with the hard work of thesis.

One of the most important thanks is to my mother and my brother. They were always with me. Especially, my mother, Gönül DENIZERI is the most important person for this thesis. She brought me into life, educated and taught me how I should think and I thought the ideas in this thesis. If this thesis can be defined valuable, this is also her achievement.

Lastly and most importantly, I would like to express my special gratitude to my wife, Derya BIRANT. I can write too many things about her role on success of this thesis, but I will cut short. There were too many times, when I wanted to quit my work. I couldn’t continue and finish my study, without her support and insistence. Especially, at last weeks of my study, she planned my life forcible. This was not so good, but this saves my study and I am so thankful now. I prepared the ideas of this thesis, but I can’t order, write and publish it without her.

Kökten Ulas BIRANT

DEVELOPING A NEW METHODOLOGY FOR SOFTWARE PROJECTS

ABSTRACT

The research presented in this thesis is an essay of a new software development methodology. This methodology is prepared according to the agile manifesto and also accepts the accreditation obligation in market. Because of this obligation, the overall system is controlled also for Capability Maturity Model via its checklist. Another property of this management system for a software development process is that the system also advises a natural improvement path for the company from a chaotic work-flow to a disciplined and controlled system. This thesis includes the entire acceptance and the usage manual of the new methodology. So, the developers, who will try to apply this methodology in their companies, will find a complete guide for a successful implementation with process model, role definitions and documentation requirements.

Keywords: Software Engineering, Accreditation of Software Development Company, Agile Manifesto, Software Development Process, Agile Software Development Methodologies

YAZILIM PROJELERI IÇIN YENI BIR METODOLOJI GELISTIRILMESI

ÖZ

Bu tezde sunulan çalisma yeni bir yazilim gelistirme metodolojisi gelistirme denemesidir. Söz konusu metodoloji, çevik manifesto dogrultusunda hazirlanmis olmakla beraber, piyasadaki yetkilendirme zorunlulugu ve sorununu da kabul eder. Bu zorunlulugu çözümleyebilmek adina Yapabilirlik Olgunlugu Modeli üzerinden yetkilendirme çalismasi yapilmistir. Yazilim gelistirme süreçleri için olusturulan bu sistemin bir diger özelligi de uygulayici sirkete düzensiz bir çalisma yapisindan daha kontrollü ve disiplinli bir sisteme dogru geçerken bir rota tavsiye edebilmesidir. Ayni zamanda tez içerisinde yeni metodolojinin onaylama raporlari ve tüm uygulama altyapisi mevcuttur. Böylece, bu süreci sirketinde uygulamayi düsünecek gelistiricilere de süreç modeli, rol tanimlamalari ve raporlama istekleri ile birlikte basarili bir uygulama için gerekli tam bir kilavuz ortaya konulmustur.

Anahtar sözcükler : Yazilim Mühendisligi, Yazilim Gelistirme Sirketinin Yetkilendirilmesi, Çevik Manifesto, Yazilim Gelistirme Süreci, Çevik Yazilim Gelistirme Metodolojileri

CONTENTS

Page

CHAPTER ONE
INTRODUCTION

1.1 General

Software development is hard to classify, whether it is a job or an art. These two complimentary opinions are partly true. The development process of software can be defined either as a product, because of the usage areas and engineering process in development cycle or as a work of art, because of the creation process in development cycle. So, these two definitions are not false, but also not true.

When the software is seen as an assistant tool for the companies in several works, as a product to earn money or as a component-based thing, which can be produced by the parts from the similar and previously prepared products, then it may be named as a product of an engineering job.

When the software is seen as an assistant tool for the research and development processes to accelerate, as an application of the new ideas or as a unique compilation for specific needs of users, then it may be named as a work of art.

The important problem begins here. For which name of software should a solution be developed, for a software “product” or for a software “art”? Software Development Methodologies try to solve the development problems of these software “things” via formulating the process.

This thesis is an experiment for a new software development methodology. According to the needs of software market, an agile software development methodology, which bases on Extreme Programming, is defined. One of the important additions to the existing agile development methods in market is the maturity support to the company. This maturity support is provided via the Capability Maturity Model from Software Engineering Institute of Carnegie Mellon University and changes according to the parameters of company.

1.1.1 Software Engineering

Software Engineering can be defined as the engineering process for software. By expanding this definition, the meaning of the software should be known. Software is a complete product of an engineering process by developing a program. In other words, software includes the program and whole documentation, which can be used for development, for maintenance or for easier usage.

The first definition problem comes here into account: “Program or Software?” “Program” is an autonomous piece of code that could be executed to solve a problem, which is computerized. “Software” is a more general name than program and includes “program” only as a part of the product. Other parts of software may be listed as specification document, design document, accounting documents, project management documents, all types of manuals.

As a result, it can be said that software engineering is an engineering discipline for defining processes to develop such a product.

1.1.2 Software Development Methodologies

Software development methodologies are the important results of Software Engineering studies. The problems are defined as the faulty estimation, low quality and not measuring the product and the development process. One of the solutions for problems in the software market is these methodologies.

A software development methodology is a formulization process of the successful development. Software Engineering branch tries to formulize the successful systems and apply them to other software developer groups. (The application process should be thought according to the several parameters of development group, like size, project type, experience, etc.)

First try of the Software Development methodologies comes from the other engineering practices. These practices force the engineers to run on a standard plan. The plan often begins with a phase to understand the needs and domain of the customer. After defining the needs, the engineers should define the solution. The engineers define their solution with the reasons of the critical decisions on paper and discuss them to find the best and correct way. The implementation of the solution begins after defining every possibility and the reasons. Each phase should be well-documented and the maintainers of the product will correct or make the software better according to these documents. (Sommerville, 1992)

The software development methodologies include these steps, because these phases are similar for a good product. However, they are not the same, because each engineered product should have different characteristics according to many different parameters. The meanings, the lengths and the orders may differ from methodology to methodology. (For example, heavyweight methodologies follow the defined system in a disciplined and documented way. However, lightweight methodologies decrease the lengths of the phase and the orders majorly when compared with the heavyweight ones.) (Pressman, 2001)

1.1.3 Main Points of Base Ideas

1.1.3.1 Why Agile Software Development?

The production (and/or development) methodologies of base engineering products are the inspiration of software development methodologies and there are not so many methods to produce a product in a regular way. When it is thought to prepare a product disciplined and controlled, the methods have the same phases; defining requirements of customer, designing the product, implementing and maintaining in market. So, phases of agile software developments are similar to heavy ones. But, the agile ones have some inner differences to the other ones and these differences obtain some advantages to these methodologies.

Agile software development methodologies obtain a faster development to the companies than the classical methods. This faster development is so important for the companies, because the technology changes also rapidly. Another rapid change will be seen also in the customers needs. Because of these rapid changes in the software development domain, the methodology should act according to these changes.

Another important advantage of the agile methodologies is the appropriateness to the own unique properties of the software. The unique properties of software and software developers are disregarded in the heavyweight methodologies. However, in agile software development methodologies; the art and job dimensions of the software are thought and the application was done for a faster production of software without disregarding the creation (art) property.

1.1.3.2 Why Capability Maturity Model?

The first question to ask may be the reason of a software development model. The software development methodologies try to ensure the quality of the product. However, the development process and the guarantee about the success of selected software development methodology are also important.

First reason to prepare such a model may be seen as the value of the market. The software is one of the most valuable and expensive products in the market. When it is thought that the resources of this product are new ideas, time and experience; then the value of the product may be seen acceptable. And the customers should be more deliberate, if they purchase so expensive things.

The software market has many companies, who can do similar things. The products of these companies may be seen same from outside. However, there are more parameters than the interface or functionality to think before signing a contract with the development company. (For example; the ability of completing the product at the right time, with the right resources, etc. may be seen in the criteria.) The biggest problem here is being sure about the success of the project, before it begins.

The software engineers tried to prepare a model to determine a company, whether it is safe or not, because of these reasons. The Capability Maturity Model is one of these models. The model recommends defining a company as mature, if the company may get the required point from the defined checklist. (For example; the documentation archives about the previous projects of the company will be used to determine the mentioned point.)

Capability Maturity Model was chosen in this thesis. First one of the reasons is the reputation of the model. The model is used in several companies in several countries. Additionally, the customers, who spend much money for software, want Capability Maturity Model accreditation for an approval before the project begins.

The Maturity Model is a product of Software Engineering Institute of Carnegie Mellon University. The Institute works on the model with many surveys in the market and with a great experience on Software Engineering. Because of this preparation method, the model may be called as one of the trustworthy models to approve the maturity of the development company and development process.

1.2 The Purpose of the Thesis

There are some software development methodologies in the market, which are used in the companies. And one of the major problems about the methodologies is specifying a target group from the software development team. For example, some methodologies focus on the problems of management and try to solve the problems for better management. However, some methodologies focus on the programmers and try to maximize the profits of the programmers in the process. And other methodologies focus on project executives to maximize the finance of the company. As seen in these examples, every methodology focuses on the (particular) actors in the process. But, no-one tries to understand overall process, team and development company and to maximize overall benefits.

Another problem for the software development methodologies is the stability of the chosen methodology. The creators of the methodologies are thinking only about their companies or about one type of the companies in the market. However, these companies are changing and growing organisms. And one company may use a rapid methodology, because of its speed and low requirements. But after some time, the company will begin to develop larger projects and his crew will also grow up. And at this time, the rapid methodology will be not enough for them. Yet, the system is already running and a big change in the methodology is not possible, in general. So the methodology may determine the project types (and also the future) of the company. At opposite, the same problems may be found after selecting a document-based, highly-controlled methodology. Here, the main problem is "The methodologies don't think over changing and improving parameters of the companies."

1.3 Thesis Organization

The thesis consists of 4 chapters.

Chapter 1 presents the general information about software engineering. Firstly, the need of software engineering and advantages of software development methodologies are explained. After that, the purpose of accreditation and popular accreditation methods for software development companies are defined. With the motivations behind this thesis are written at last part of this chapter.

Chapter 2 begins with details of Software Engineering methodologies and the trendy approach: Agile Manifesto. Extreme Programming is described to form base information about an agile software development methodology. After that, new software development methodology, RINGS is defined. The methodology is explained detailed with its process steps, team roles, documentation requirements, advantages and risks.

In Chapter 3, the acceptance of RINGS is discussed. Two methods are used for accreditation of RINGS. The Capability Maturity Model is described as the accreditation method and the result of RINGS for the acceptance checklists of Capability Maturity Model is in that chapter. Secondary acceptance method used in this thesis, is the acceptance of market. In this chapter, the opinions of possible users of RINGS are also defined.

In Chapter 4, the overall summary and possible future works were explained and a conclusion about the study can be found.

As Appendices, some documentary about the success story of Extreme Programming and the questionnaire, which is applied to project managers and software developers for gathering their opinions about “RINGS”, may be found.

CHAPTER TWO
RINGS

2.1 Overview

“Rings” is a development methodology for software and company, which totally accepts Agile Manifesto. The methodology supervises not only the success of the software development project, but also the improvement of the company according to the changing parameters.

Rings methodology is formed like 3 rings, to define different software development processes and easier transitions between these processes according to changes in parameters of company. (Like alteration of the types of the projects, size of the project teams, requirements of the new projects, etc.) The changes of the parameters cause also to change the ring used for project management.

“Rings” is a complete process definition with 3 parts. These 3 parts are the process steps, role definitions of the developer team according to the different rings and the documentation requirements according to the process steps.

The backbone of the developer team is programmers, which has the knowledge about the basic software development process steps. (Not only programming, but also requirement elicitation, design, testing, etc.) At first ring, the team should have also a project manager. The company at second ring should also define two more roles in the team: Surgeon and Reporter. At the third ring, the company should also hire a quality manager.

The documentation requirements of project steps increase also ring-by-ring. At the first ring, the documentation requirement is limited. The documents, which are produced by the project development phases, increase by changing rings. (This change is parallel with the complexity of the project and the possible problems of management of the bigger development team.)

“Rings” has 4 key principles. These principles are the motivations behind “Rings”. The developers should think about these ideas, while deciding something about the software project.

In the following sections, the basic concepts about a software development methodology may be found. In last sections of the chapter, the details of principles, project process, team and document requirements will be detailed.

2.2 Related Works and Basic Concepts

2.2.1 Basic Life-Cycle of Software Development

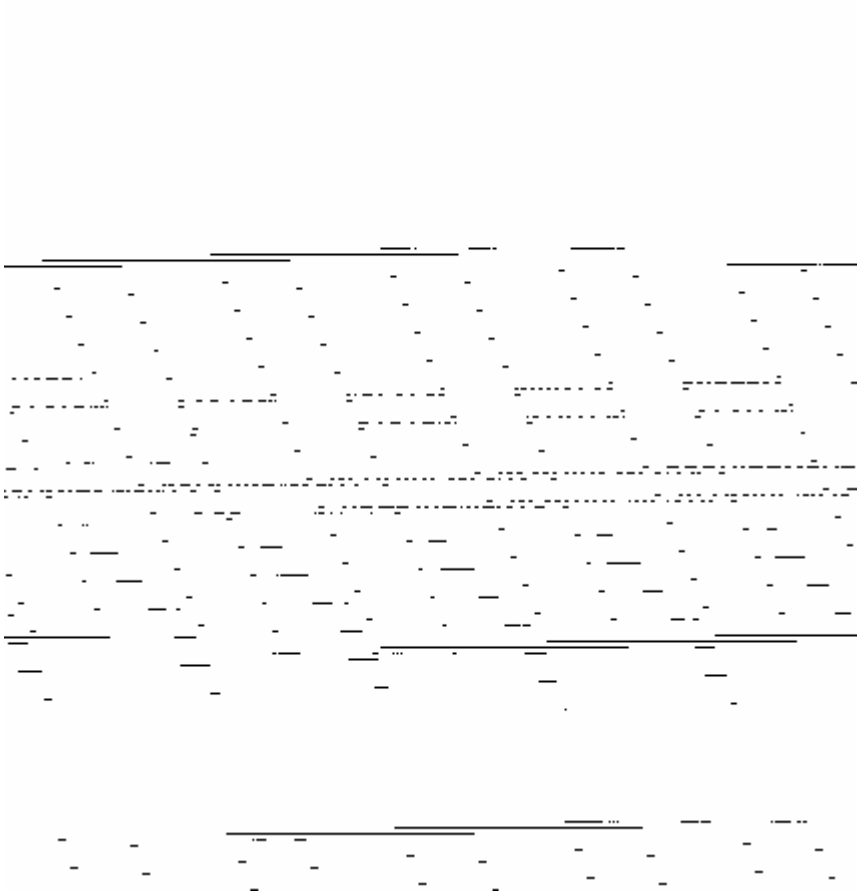

Figure 2.1 Waterfall model of W. Royce

Software is a product, which is developed via an engineering process. After 1961, NATO Conference, it is said that the solution of the problems on software production is preparing software development process over a basic life cycle. According to the experiences on other engineering disciplines, the engineers and software builders are prepared a basic life cycle, which will be the base for developed processes. (Ghezzi, Jazayeri and Mandrioli, 1991) (Schach, 2002)

The basic life cycle starts with the requirements analysis of the software and tries to define the software production as a project. This project will also die with the retirement step of the basic life cycle: (Royce, 1970)

Requirements Analysis: At the beginning, only the customers know what they want. The transmission of this knowledge is the first problem in context. The developer team should try to get the requests of the possible users and customers at first. Next process is the analysis of these requirements according to the abilities and for the highest profit. (Sommerville and Sawyer, 1997)

Requirements Specification: The collected requirements will be refined with the whole development team to concretize the product with the requests. This stage is also called “What to do?” – Stage. The signed specification document can be thought as an indirect acceptance of the customer and developer about the main topics and general boundaries of the product.

Design: The development team will try to design the overall product according to the specified requirements. By design phase, the developers try to specify all of the technical definitions to think about the problems, before implementation. (Because it can be said that the earlier founded syntactic or semantic errors always costs less.) This phase can be also called as “How to do?” – Stage.

Implementation: The designed software product will be implemented after specifying and solving almost all of the design problems. Implementation phase can be explained shortly as coding the prepared design via a programming language.

Although some researchers think that testing of the implemented codes should be defined out of the implementation phase, the testing work will be done in the implementation phase.

Maintenance: After implementation, the product will be send to the market or to the unique customer. And some changes may be required to correct or improve the product. These changes are thought as maintenance phase of a standard software life cycle.

Retirement: The last phase of a standard life cycle is dying, which can be seen as a metaphor for software product. When software's maintenance is more expensive than preparing new one, it can be said that the software enters to the retirement phase. And some documentation work about abilities of the existing system will be begun to facilitate nest process.

After preparing these basic milestones of software production project, the software engineers tried to work on these components in the life cycle to be more efficient. Some stages were called as less important or some were called as iterative stages for specified systems. And these implementation changes are defined as different software development processes. Waterfall, Spiral, Incremental, Extreme Programming, SCRUM, Crystal, etc. may be seen as the most popular processes in software development history.

According to the base manner, these development models can be classified into two classes:

Classical models (or Heavyweight models) include the Waterfall (Royce, 1970), Boehm's Spiral (Boehm, 1988), and Incremental models. These models are generally documentation-based and pay more attention on management rather than product development. The motivation behind this is that the better management brings naturally better result for development. These models are being used from the beginning of the software engineering history.

Agile models (or Lightweight models) include the Extreme Programming, Crystal, SCRUM, etc. These models are generally product-based models and pay more on rapid customer satisfaction rather than long-term works. Because of this, these models especially can't be used widely for large projects. The motivation behind these models is the rapid changing world and conditions in the market. The one of oldest models from agile manifesto can be said after 2000s. In the section, Agile Methodologies and Extreme Programming will be detailed, because the main motivation of this thesis is agile models.

2.2.2 Agile Software Development Manifesto

After 1961 NATO Conference, the software engineers tried to model the software development process to solve the main problems of developers and customers. And they used other engineering disciplines as base methodologies. Because the products of other engineering disciplines are not similar with software development, these imported solutions couldn't solve the problems. (It can be also said that these methodologies delayed the solution, because other engineering disciplines are suitable for the special situation of the software development operation. The product, the producers, the planning phase, the experiences, etc. are so different from classical engineering worries.)

Especially the engineers from large companies tried to change the classical management-based systems according to the nature of the software development. In the second half of 90s, some software engineers begun to implement their "light" ideas in their companies and the success stories were also begun explaining in all over the world.

Via internet, these success stories and founders of these ideas were begun to compare. The software engineering researchers and the founders of these models are decided that these models can be classified with some attributes against the classical

models. To specify these common properties, some software engineers were come together in UTAH.

In a meeting at Wasatch Mountains (UTAH), 17 project leaders/software developers specified these common properties. At February, 13 2001, these software engineers signed the Agile Manifesto, which contains the common properties of the trendy development models. The agile manifesto contains some sentences for software development to be successful in the new software market and has some contrary ideas:

"Agile Manifesto:

We have come to value;

"Individuals and interactions" over "Processes and tools",

"Working software" over "Comprehensive documentation",

"Customer collaboration" over "Contract negotiation",

"Respond to change" over "Following a plan".

That is while there is value on the items on the right; we value the items on the left more." (Agile Alliance)"

The idea behind the manifesto is the changing market requirement of the changing world. The first rule is that the software development is a work of human and cannot be fully automated with process rules and tools. Second attribute is that the first condition of customer satisfaction is the working software and too detailed documentation for management is unnecessary. The development team must have collaboration with customer, because the customer will pay for the product. As a last rule, the engineers said that always the conditions will change during the development and the development team can't resist these changes for following a plan.

According to this manifesto; the developers accept these following principles;

"We follow these principles;

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

b. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

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

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

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

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

g. Working software is the primary measure of progress.

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

i. Continuous attention to technical excellence and good design enhances agility.

j. Simplicity -- the art of maximizing the amount of work not done -- is essential.

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

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

(Principles of Agile Manifesto, n. d.)

The software developers, who signed this manifesto has their own famous development methodologies. However, one of these methodologies becomes the

most famous one with its easy implementation and high understandability: Extreme Programming.

2.2.3 Extreme Programming

Extreme Programming (XP) is the most famous agile methodology. When the history of the Extreme Programming is investigated, Daimler Chrysler can be seen as the first implementer. Two engineers from the software group, Kent BECK and Ward CUNNINGHAM implemented their original ideas about development cycle in 1996. The unexpected, successful results of this implementation were spread all over the world. With the help of the new technology, internet, the success story of the Chrysler has become a solution for the software world. The details of C3 project of Daimler-Chrysler may be found in Appendix A.

2.2.3.1 Four Values of Extreme Programming

Extreme Programming has four values and 12 implementation practices by definition. Especially the four values of Extreme Programming are the physiological differences against the classical methodologies: (Beck, 1999)

Communication: The communication for a software development is very important for a successful development and successful software product. The communication has two meanings here: The communication between user and developer is the first idea of communication. The idea is that the classical development model requires distinct phases for communication (only requirements and specification phases) However, XP requires continuous communication with user/customer. The second communication is between developers. Usually the management or documentation based systems pays more attention to the management and for easier management, the developers only communicate via tools or reports. However, XP advocates that human can better communicate by speaking rather than reports.

Simplification: The software development process can be seen as a correction process. Because of the requirements, specification and market problems, the developers should change the prepared software all over the time. The motivation is that the developers can't resist to the changes during the development. And because of this, it is not effective to develop more complicated algorithms like an artist. The developers should prepare their algorithms as simple as possible to decrease the change effort and also to decrease the spend effort for change-able code.

Feedback: One of the most important problems on the software development is the mismatch of the real-requirements on the delivery. To solve this problem, the testing mechanisms are tried to improve by classical methodologies. However, XP says that the real problem begins from the motivation. As a testing mechanism, the developers might use the real users by giving the incomplete systems and taking feedbacks from them. These feedbacks are more efficient than the automated testing tools or more realistic than the stakeholders with customer role.

Courage: The last motivational value of Extreme Programming is being courageous. The courage should be in each phase of the development. The developers should be always in communication with the users. And if they have fear for doing false, they can't talk with the user or they can't show the system to the user. This will postpone the detection and correction of the errors. Or when the developers are not courageous, the developers cannot try some revolutionary design changes. This will cause the destruction of the creativity of the developers.

2.2.3.2 Development Process of Extreme Programming

The development process of Extreme Programming is not so different from the classical models. Like other methodologies, Extreme Programming can be also defined with process steps. (Wells, 2006)

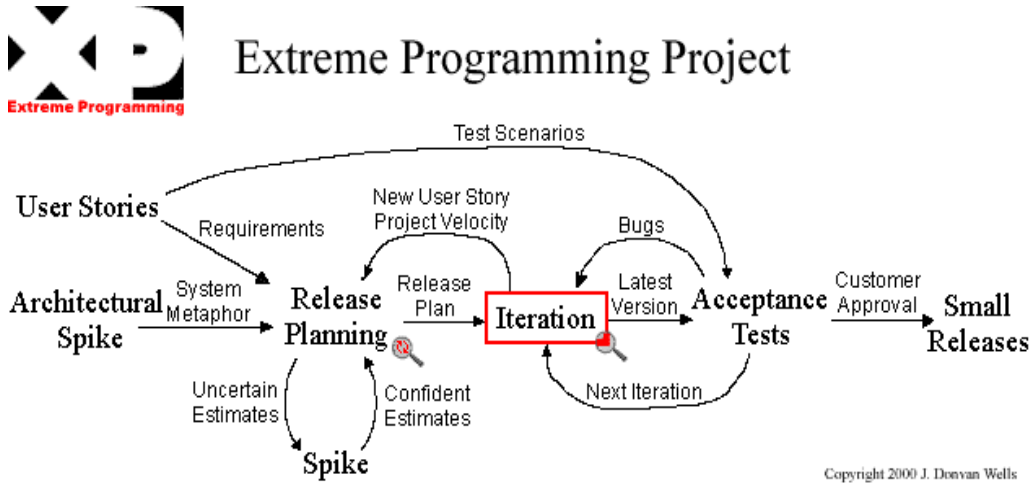

Figure 2.2 General development cycle of an XP Project

The overall process may be seen like incremental development. For one project, the process begins with User Stories as seen in Figure 2.2. With inputs from user stories, system metaphor and estimations, the release plan will be prepared. After that, according to this release plan, the iterations begin. Acceptance of iterations will be judged with acceptance tests. When the iteration is verified by acceptance test, the prepared iteration is integrated into the release via a customer approval.

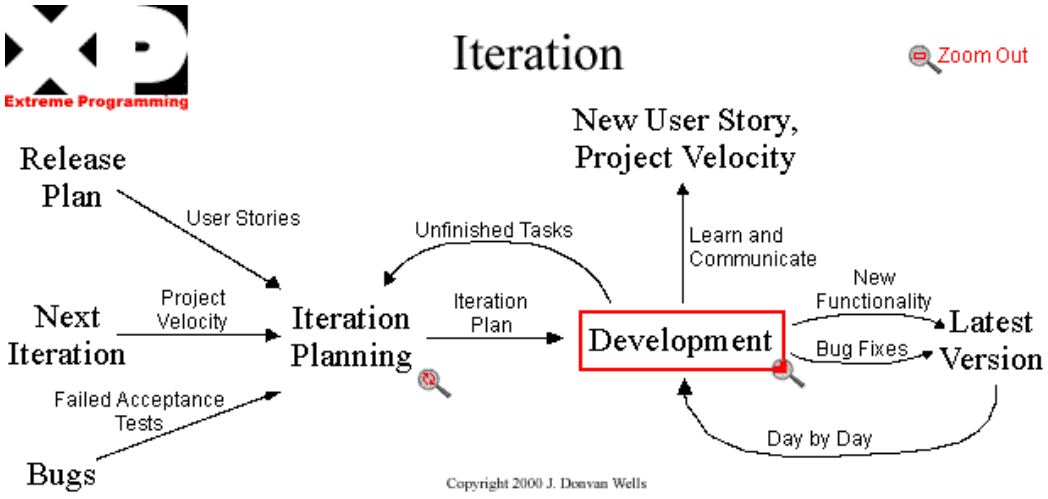

Figure 2.3 Iteration of an XP Project

When iteration should be detailed, it can be seen that iteration is a small project prototype. As seen in Figure 2.3, iteration begins with a planning phase via a small risk analysis with inputs from release plan and bugs from previous iteration.

According to the details of the iteration plan, development cycle will begin. When some problems are defined about the user stories or requirements, new user stories or updates on existing ones may be gathered from users. Day by day, the outputs, new functionalities from development will be added to the latest version. The prepared version will be the output for overall project.

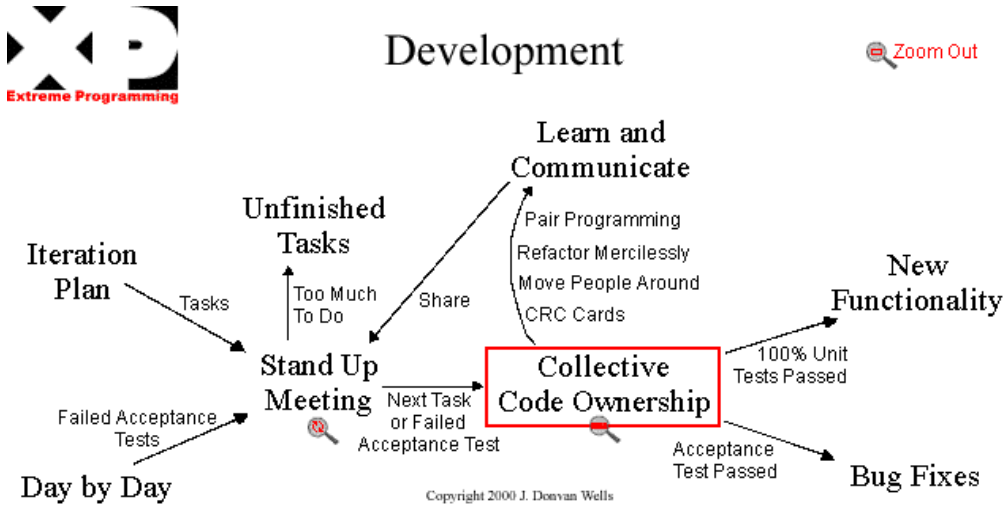

Figure 2.4 One development cycle of an iteration of an XP Project

After iteration plan is produced, the coding phase of the development may start. (Figure 2.4) The test results and plan are the inputs of the inner - design meetings. These meetings may be called as limited design phases. These daily meetings are for determining and analyzing the tasks and after that the coding phase will begin.

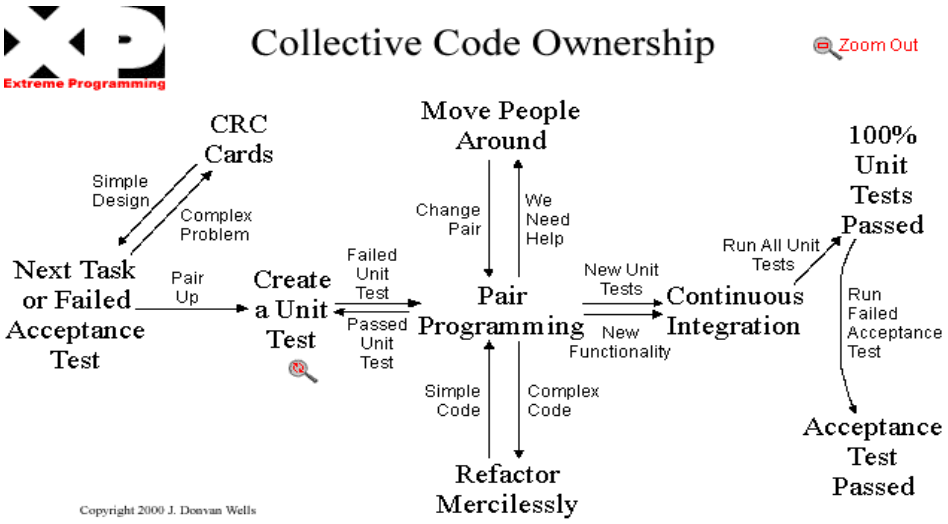

Figure 2.5 Coding phase of an iteration of an XP Project

As seen in Figure 2.5, each coding phase include too many coding ideas. Coding of new task will begin always with preparing the unit tests for new task. After that, the coding session will be done according to pair programming. Other programming practices for better development are refactoring and continuous integration.

2.2.3.3 12 Practices of Extreme Programming

Extreme Programming works according to 12 practices. These 12 practices are defined with 4 values and the base of the process. While some of these practices are new terms, some of them are already defined, but not experienced software engineering ideas. The 12 practices of Extreme Programming are; (Fowler, 2005; Jeffries, 2001)

1. Planning Game: The Planning game will be played with users to take their requirements with their own technical terms. This may be seen the requirements elicitation phase of Extreme Programming.

2. Small releases: The system will have small releases and the user will see what he will get in small time periods.

3. Metaphor: The system should be defined with metaphors for the developers for better understanding. The target system should be defined in a simple shared story as a guide.

4. Simple Design: The design should be as simple as possible to gain the change-ability. Extra complexities may be useful to increase the functional performance, but the cost of it may be too much. So, the simple design will decrease the maintenance and correction costs.

5. Testing: The testing is too important and the test cases should be defined before the implementation phase. (If we will know what we will test, we can also code it

right) Programmers continually write unit tests, which must run for development to continue. Customers write tests demonstrating that features are finished.

6. Refactoring: The developers should be courageous and not to fear from refactoring the system. The development cycle should include the phases to make the design, functionality and performance better.

7. Pair Programming: The coders will write the code as pairs. Two developers, on one module and one computer will show better performance than two developers on two modules. (Birant and Kut, 2004; Cockburn and Williams, 2000; Ferzli, Wiebe and Williams, 2002)

8. Collective Ownership: The whole development team owns all of the produced code. We cannot divide the code into pieces to specify responsibilities. So, anyone can change any code anywhere in the system at any system.

9. 40-hour week: The developers are responsible to design code and test. And because of this hard work, they should not to be over-worked. (Or as a more realistic advise; never work overtime a second week in a row.)

10. On-site customer: The customers should attend the development team and if possible, he should attend the development process full time. The practice advises that, because the most important and true information source about project is the customer and if the time decreases to reach the source, it will directly affect the development time.

11. Continuous Integration: The system will be developed with pieces; however the whole product should be prepared incremental. This means that the system should be expanded with completed parts. Previously thought methodologies don’t think over the integration problems. However, the development team should think over the integration phase in Extreme Programming and should meet these problems as early as possible.

12. Coding Standards: The developers should have some coding standards to increase the readability of their codes, because collective ownership and pair programming forces all development to understand each others code. Another advantage of this practice is also providing the agility while team changes so often in a company.

2.3 A new methodology: RINGS

2.3.1 4 Values of RINGS

The software development is a work between art and job. For defining the place, first of all, the developers should have the same ideas and should be agreed on the same principles:

Communication: The communication in the group is too important. There are two types of communication in a development process. The communication between two developers is important, because human being is a social creature. The information transfer is not so easy in other ways rather than talking. So, the communication should be clear and easy in a development process. The communication between developers and customers is also important, because the only input about the product is customer. So the information transfer from the customer to developer is important for preparing the right product right.

Coordination: The software development is a job which has its own properties and the software developers just want to feel themselves free. However, this doesn't mean to leave them free. The coordination via a coordinator is as important as being free. There should be a coordination manner to coordinate the system, communication and development process.

Courage: The courage of the developers is also important in a development process. The courage may be required by a discussion with the customers. The

developers should give the prepared part of the software to the customer to take the feedback without fear. In another case, the developers should have the courage to delete the wrong code of the software. The courage of deleting something instead of trying to repair is too hard, but it may be necessary in some cases.

Continuous Coding: The software developers or (in general) computer engineers don't like to write reports. Documentation is always a problem in software development groups. In our method, the developers are needed not to write documents. Instead of this, some solutions will be explained in the process. However, the principle is not to spend time to write reports for management level.

2.3.2 Overall Process of Rings

"Rings" is a development methodology for software and company, which totally accepts Agile Manifesto. The methodology supervises not only the success of the software development project, but also the improvement of the company.

Rings methodology is formed like 3 rings, to define different software development processes and easier transitions between these processes according to changes in parameters of company. (Like alteration of the types of the projects, size of the project teams, requirements of the new projects, etc.) The changes of the parameters also cause to change the ring used for project management.

"Rings" is a complete process definition with 3 parts. These 3 parts are the process steps, role definitions of the developer team according to the different rings and the documentation requirements according to the process steps.

The backbone of the developer team is programmers, which has the knowledge about the basic software development process steps. (Not only programming, but also requirement elicitation, design, testing, etc.) At first ring, the team should have also a project manager. The company at second ring should also define two more

roles in the team: Surgeon and Reporter. At the third ring, the company should also hire a quality manager.

The documentation requirements of project steps increase also ring-by-ring. At the first ring, the documentation requirement is not many. The documents, which are produced by the project development phases, increase by changing rings. (This change is parallel with the complexity of the project and the possible problems of management of the bigger development team.)

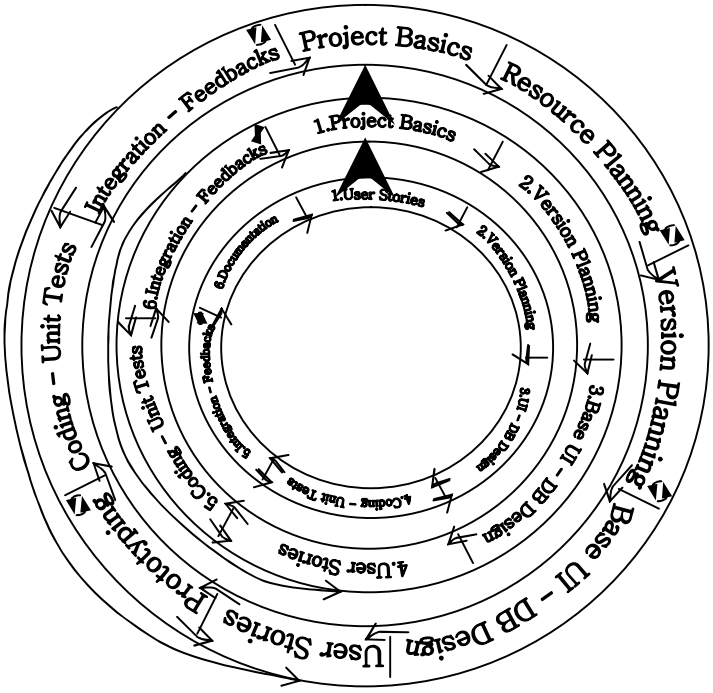

Figure 2.6 Overall process of “RINGS”

“Rings” has 4 key principles. These principles are the motivations behind “Rings”. The developers should think about these ideas, while deciding something about the software project. These mental properties are explained in next chapter.

2.3.2.1 Inner Ring

The start-up point of the process is a young software development company with start-up projects. And the first development process for this company is “Inner Ring”. The prudence is that the company has not so much resources and the projects are

small-sized or medium-sized. So, only a project manager role may be enough in this ring. In spite of the many assignments of the project manager in other rings, the project manager of inner ring may work only for coordination and as result; he also may help to develop as a coder.

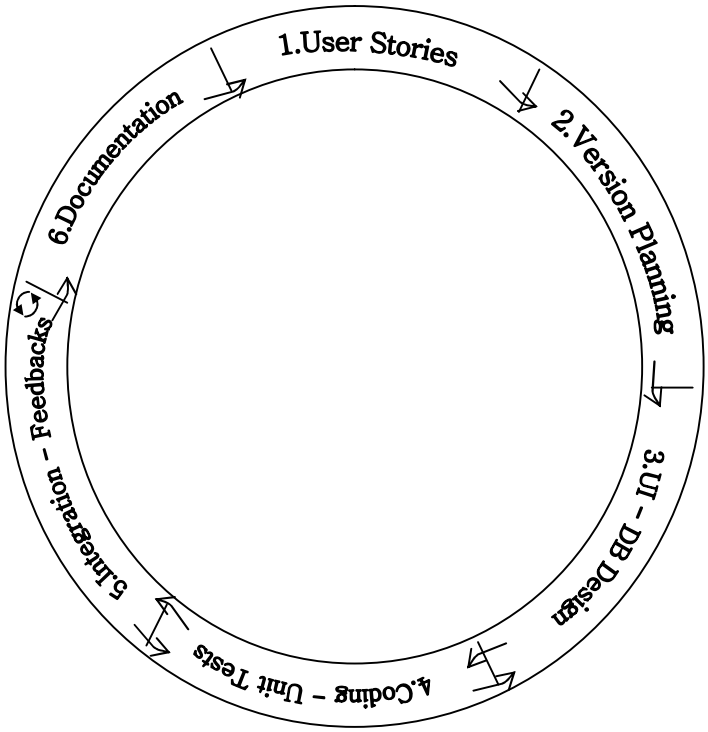

Figure 2.7 1st Ring of "Rings" (Inner Ring)

As seen in Figure 2.7, the process begins with requirements elicitation (User stories). The attendees of the meeting will be the project manager, project hero (if exists) and customer representatives. The stories of customer representatives will be noted and a general database design and user interface design will be done. According to the experiences of the hero and the manager, a release plan will be published to the group and the development process begins. Before the programming phase begins, the release plan may be the only printed document of the system. After the functionality and the data storage is accepted by the customers, the functions of the user interface will be written by the programmers. While coding, the unit tests are also applied to the prepared codes and a continuous integration will be done. The most important testing phase is taking feedbacks from the customer. While continuous integration, the prepared codes are sent to the customer (or to the

customer representative) and their feedbacks are taken as test results. After all of the coding is done, the required documents (like manuals, code definitions, etc.) will be collected from the development environment. When focused on each of the process steps;

- *User Stories:* User Stories are the requirement elicitation method of a “Rings” process. These stories are not written formally, because the users should let feel as free as possible by defining their needs. The formats and obligations for not using terminology prevent the user from explaining the needs as good as possible. So, user stories may be seen as the functionality description of the system according to the terminology of it. The project manager and the developers will be in this step of the process to define the requirements of the system. They take the real functionality description according to the real terminology of the system from the real users of it. The General Specification document is the result of this step. This document includes the boundaries and validation criteria of the software project.

- *Version Planning:* Version Planning may be seen as a primitive study of time-estimation or schedule-planning. The Version Planning term is defined in Extreme Programming. According to the base definition; the system should be represented to the customers in short time periods. Each representation should include new functional or non-functional innovations in software. These representations are called as versions and the definitions and estimations of the versions have two advantages: (I) The customers are informed about the future of the software project. (II) The developers have a time constraint to follow. So according to the user stories, the project manager will define the production speed, the order of the functional or non-functional requirement implementations and the representation schedule. So, the version plan may be seen as a transition to the schedule planning. As a result of this step, the version plan document, which include the written conditions, should be prepared and accepted by all of the developers.

- *User Interface – Database Design:* One of the reasons of designing the software before implementation is need of a written thing to discuss the implementation. After the design of a product is prepared on a paper, the chance of satisfaction of both the developers and the future users increase. The developers may discuss easier over the solutions (for example; algorithms, data types, etc.) of the software, when a study of solution is ready. Also, the users of the software are trying to define their requirements. However, after defining some requirements if the developers represent some concrete solutions according to the defined problems; the users may see easier, whether his definition is what he wants or not. So, the user interface design may be seen as the concretion of the user requirements to work on them. Also the Database Design of the system is the study of preparing the centre of the project. According to these base designs, the developers may easily discuss the implementation of the software.

- *Individual Development:* The product of the development team, in 1st ring will be a small project. So, the team is not so large. Because of this, a detailed and iterative development (implementation) is not considered. The implementation of one-module product will be according to the following process steps;

    - *Coding – Unit Tests:* The coding phase is the easiest phase step of "Rings" to define. The simple programming of the software is the meaning of this step. In this phase, pair programming may be found useful and may be used. But, the human resource of the project is significant. If the human resource is enough, pair programming is preferred. While coding, the unit tests will be also applied to the inner units (Or functions) of the system to verify it.

- *Integration – Feedbacks:* Continuous integration is a meaningful practice. To specify the problems of product, the errors of the communication between the units is as important as the errors of the units. So the produced units should be integrated as often as possible. To validate the project, the feedbacks are too important. “RINGS” accepts that the most important information source of a software project is the user of the product. So, each version will be delivered to user to take feedbacks about the verification and validation of the project.

- *Documentation:* The documentation is the most important thing for a well-controlled, disciplined and measured software development process. Software development process is a production process of an invisible product and the documentation tries to let the product visible for both developers and customers. However, preparing formatted documents are also most frightening nightmares for software developers. The software developers always want to focus on programming, not on preparing documents before or after programming. Especially, preparing documents is harder in small projects, because it is harder to let the developers believe in importance of documentation. But, the managers should motivate the developers to prepare documents during development for easier maintenance and preparing a better base for future products. Yet, the actuality is not so easy. Besides motivating the developers and also forcing them, may be not enough. And documentation during development is only a utopia, especially in small projects. Inner ring advises accepting the reality. The documentation phase may be shifted to the end of the development process. So, the developers may focus only on development and also the customers and the project managers will gather the required documentation. The documentation phase will start with end of the implementation of program and will end with production of documents. (Additionally, the requirements of documentation phase should be as minimal as possible.)

2.3.2.2 Transition from Inner Ring to Middle Ring

The “Inner Ring” is not a complex process. It may be used only in small projects. And by time, the company will work on medium-sized projects. So, the “Inner Ring” will be not enough for these projects. (For example; in a small project, documentation about prepared codes is not so important, because the prepared codes are not so many. However, when the line of codes increase, the information to remember also increase and the developers should be more documented.) For these projects, the company will need a more-disciplined and controlled process. Because of this, the company should change its “ring”.

While changing the process cycle, the company should be prepared. The company should prepare its “Coding Standards”. This manual of standards are defining the abbreviations, naming strategies and commenting of the coding group and it will be helpful for more complex projects. When the team will change so often and when enlarge more than a manager can coordinate, this standard will be helpful. The adaptation time of the new comers will decrease with the help of this manual. And with this manual, the company may start using “Middle Ring” process.

2.3.2.3 Middle Ring

After 2 or 3 projects, the cycle will be not enough to develop good projects, because of the increased project number and maintenance requirements. At that time, the company should go into the 2nd circle of the cycle. This circle contains more documentation than the inner one. The requirements engineering part is divided into two parts. In the first part, the general knowledge will be collected from the users (maybe from the customers) However, in the second part, the deeper information will be collected from the real users of the software by the developers. This will increase the success chance of validation process. And some more changes will be done in the company, which are not so hard to adapt.

Because the process requirements are increased, the roles of the team should also increase. First of all, the project manager should only work for coordinating the system. Additionally, one of the software heroes of the company will be assigned as the surgeon of the project. Two more management group members will be added to the list: A documenter and a project management tool (software)

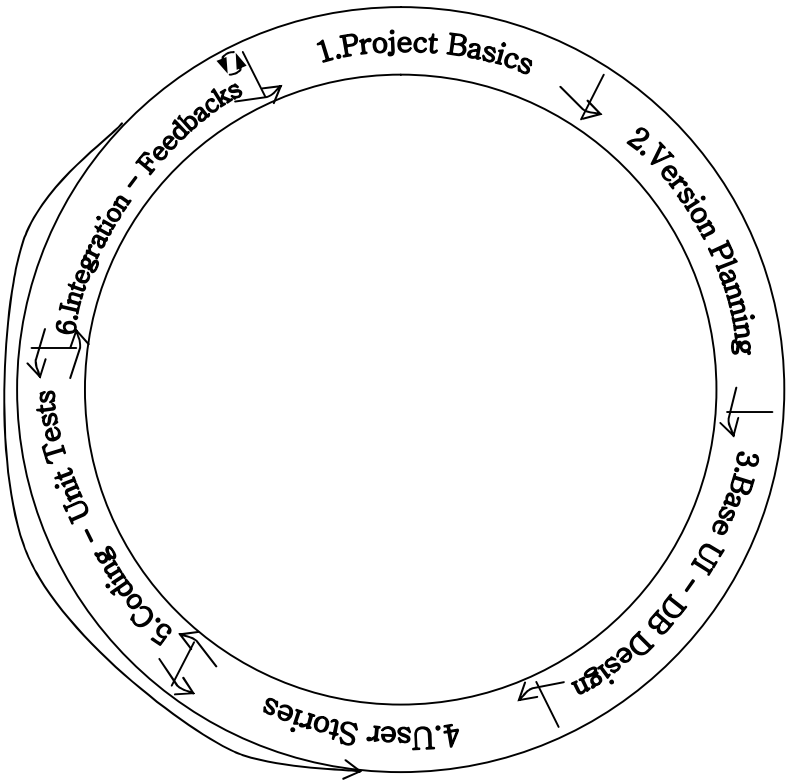

Figure 2.8 2nd Ring of "Rings" (Middle Ring)

The overall process begins with a restricted requirements elicitation step (Figure 2.8). The first part of requirements engineering is the starter of the project. In the first meeting, the management group of the project will prepare the boundaries and the overall scenario of the software. This meeting will be done with Project Manager, Surgeon and Customer Representatives. After the release planning, the base of database system and user interface of the software will be decided. After that the parts (modules, components or functionalities) of the software project begins. At each module, the developers run a requirements engineering process by themselves and try to code the taken requirements. The implementation phase of the cycle may

be seen as incremental development. The implementation details of each process steps may be defined as follows;

- *Project Basics:* Project Basics is the first step of the project definition. The management team and the customer representatives will be met on system and discuss the basic boundaries of it. According to the level of experience of surgeon and project manager; the basic requirements definition may be done in this step. The collected information should be enough to define the version plans, developer requirements, modules (component/sub-system/etc.), time requirements, basic performance requirements, legal requirements and financial costs. However, the functional or non-functional details of the separated modules will not be defined in this step. The “Project Basics” will end with preparation of the general specification document. This document will be the overall guide for defining the boundaries of the project and as a restrictive document for asking for new needs in future steps of project.

- *Version Planning:* Version Planning may be seen as a primitive study of time-estimation or schedule-planning. The Version Planning term is defined in Extreme Programming. According to the base definition; the system should be represented to the customers in short time periods. Each representation should include new functional or non-functional innovations in software. These representations are called as versions and the definitions and estimations of the versions have two advantages: (I) The customers are informed about the future of the software project. (II) The developers have a time constraint to follow. So according to the user stories, the project manager will define the production speed, the order of the functional or non-functional requirement implementations and the representation schedule. So, the version plan may be seen as a transition to the schedule planning. As a result of this step, the version plan document, which include the written conditions should be prepared and accepted by all of the developers.

- *Base User Interface – Database Design:* One of the reasons of designing the software before implementation is need of a written thing to discuss the implementation. After the design of a product is prepared on a paper, the chance of satisfaction of both the developers and the future users increase. The developers may discuss easier over the solutions (for example; algorithms, data types, etc.) of the software, when a study of solution is ready. Also, the users of the software are trying to define their requirements. However, after defining some requirements if the developers represent some concrete solutions according to the defined problems; the users may see easier, whether his definition is what he wants or not. So, the user interface design may be seen as the concretion of the user requirements to work on them. Also the Database Design of the system is the study of preparing the centre of the project. According to these base designs, the developers may easily discuss the implementation of the software. As a difference from the similar step in inner ring, there is a finalization document. The work on base user interface and database designs will be reported by the documenter.

- *Module-based (or Component-based, etc.) Pair Development:* The coding phase is thought according to the Extreme Programming suggestions. Pair Programming is advised. The overall-defined modules are detailed, designed and coded with the pairs. There may be seen an inner development ring to produce modules and develop final product with the following inner steps;

    - *User Stories:* User Stories are the requirement elicitation method of a “Rings” process. These stories are not written formal, because the users should let feel as free as possible by defining their needs. The formats and obligations for not using terminology prevent the user for explaining the needs as good as possible. So, user stories may be seen as the functionality description of the system according to the terminology of it. The project manager and the developers will be in this step of the process to define the requirements of the system. They

take the real functionality description according to the real terminology of the system from the real users of it. The General Specification document is the result of this step. This document includes the boundaries and validation criteria of the software project. For each module, the collection of user stories phase should reported with a detailed specification document. This document should be reported the work on elicitation process and specify also the details of the module according to the user needs.

- *Coding – Unit Tests:* The coding phase is the easiest phase step of "Rings" to define. The simple programming of the software is the meaning of this step. In this phase, pair programming may be found useful and may be used. But, the human resource of the project is significative. If the human resource is enough, pair programming is preferred. While coding, the unit tests will be also applied to the inner units (Or functions) of the system to verify it. Each coding step should be finished with this unit tests and the results of these test should be reported via the documenter.

- *Integration – Feedbacks:* Continuous integration is a meaningful practice. To specify the problems of product, the errors of the communication between the units is as important as the errors of the units. So the produced units should be integrated as often as possible. To validate the project, the feedbacks are too important. "RINGS" accepts that the most important information source of a software project is the user of the product. So, each version will be delivered to user to take feedbacks about the verification and validation of the project. According to the feedbacks, the development process may return to the coding phase to correct the faults or the process may go with another module and return to a new "user stories" phase. After the acceptance, the documenter will update the documents for users or future products; procedure documents and user help documents.

2.3.2.4 Transition from Middle Ring to Outer Ring

The documentation requirement of “Middle Ring” is more than the requirements of “Inner Ring”. However, increased resources force the development company to increase the management, inspection and correction mechanisms in the project. The “Outer Ring” increases these mechanisms.

When the company satisfies the 2nd stage of Capability Maturity Model, then it can be said that the company is successfully ready to go out of the “Middle Ring”. And if it is required, the company may use “Outer Ring”. Before this, the company should prepare the documentation standards and quality standards. The company should standardize the documents, because time allocated to prepare the documents will decrease and the need of the documents will increase in time. Standardizing the documentation processes helps the company.

The quality standards should be also documented and standardized. In first two rings, the time and money resources are so important. However, quality of products and quality of the company are more important for a big company. So, standards for a good quality are important. With these standards, the company may start to use “Outer Ring”.

2.3.2.5 Outer Ring

The process of the 3rd circle is too similar with the 2nd circle (Figure 2.9). First important change is the increased documentation requirements of the process, because the process supports the subcontract management and large project groups. Second important change is the more controlled process mechanism. Our second aim in the process starts here: A natural improved maturity. To achieve this aim, the management team needs to be increased by one member. The Quality Manager may be seen as the controller of metrics of process and product.

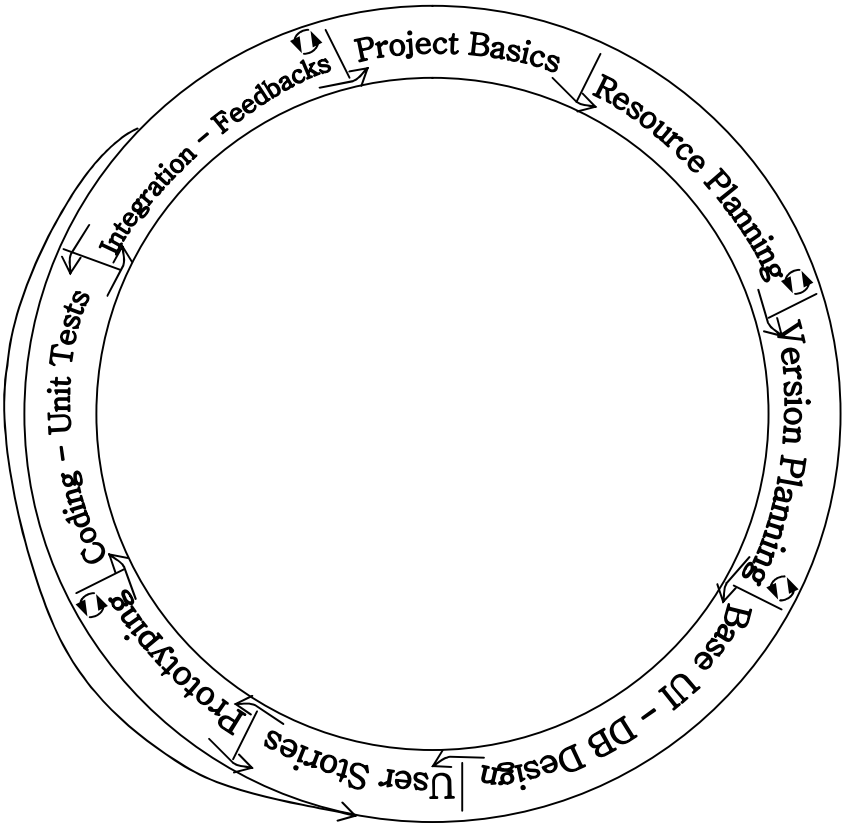

Figure 2.9 3rd Ring of "Rings" (Outer Ring)

The Management Group attends to the meeting to specify basics of the project. After this meeting, the resource plan will be prepared and discussed with the customer representatives. After corrections and acceptance, the process goes like the "Middle Ring". However, each phase has its own document requirements and review of these documents. The steps of outer ring are detailed as follows;

- *Project Basics:* Project Basics is the first step of the project definition. The management team and the customer representatives will be met on system and discuss the basic boundaries of it. According to the level of experience of surgeon and project manager; the basic requirements definition may be done in this step. The collected information should be enough to define the version plans, developer requirements, modules (component/sub-system/etc.), time requirements, basic performance requirements, legal requirements and financial costs. However, the functional or non-functional details of the separated modules will not be defined in this step. The "Project Basics" will

end with preparation of the general specification document. This document will be the overall guide for defining the boundaries of the project and as a restrictive document for asking for new needs in future steps of project.

- *Resource Planning:* The assumptions for resources, that the company may spend or that the project need started from lower level. However, in 3rd ring, the needed resources of the project will increase dramatically. So, there should be a planning phase to solve the possible problems, because there will be more resources than a project manager can handle spontaneously. The management team will work on resources and prepare an allocation plan. After preparing this plan, the customer representatives and the management team should also discuss the report until an agreement.

- *Version Planning:* Version Planning may be seen as a primitive study of time-estimation or schedule-planning. The Version Planning term is defined in Extreme Programming. According to the base definition; the system should be represented to the customers in short time periods. Each representation should include new functional or non-functional innovations in software. These representations are called as versions and the definitions and estimations of the versions have two advantages: (I) The customers are informed about the future of the software project. (II) The developers have a time constraint to follow. So according to the user stories, the project manager will define the production speed, the order of the functional or non-functional requirement implementations and the representation schedule. So, the version plan may be seen as a transition to the schedule planning. As a result of this step, the version plan document, which include the written conditions should be prepared and accepted by all of the developers.

- *Base User Interface – Database Design:* One of the reasons of designing the software before implementation is need of a written thing to discuss the implementation. After the design of a product is prepared on a paper, the

chance of satisfaction of both the developers and the future users increase. The developers may discuss easier over the solutions (for example; algorithms, data types, etc.) of the software, when a study of solution is ready. Also, the users of the software are trying to define their requirements. However, after defining some requirements if the developers represent some concrete solutions according to the defined problems; the users may see easier, whether his definition is what he wants or not. So, the user interface design may be seen as the concretion of the user requirements to work on them. Also the Database Design of the system is the study of preparing the centre of the project. According to these base designs, the developers may easier discuss the implementation of the software. As a difference from the similar step in inner ring, there is a finalization document. The work on base user interface and database designs will be reported by the documenter.

- *Module-based (or Component-based, etc.) Pair Development:* The coding phase is thought according to the Extreme Programming suggestions. Pair Programming is advised. The overall-defined modules are detailed, designed and coded with the pairs. There may be seen an inner development ring to produce modules and develop final product with the following inner steps;

    - *User Stories:* User Stories are the requirement elicitation method of a “Rings” process. These stories are not written formal, because the users should let feel as free as possible by defining their needs. The formats and obligations for not using terminology prevent the user for explaining the needs as good as possible. So, user stories may be seen as the functionality description of the system according to the terminology of it. The project manager and the developers will be in this step of the process to define the requirements of the system. They take the real functionality description according to the real terminology of the system from the real users of it.

- Prototyping: Specifying requirements is one of the biggest problems and how the project grows; this specification will become more problematic. So, to solve this specification problem, outer ring adds some additional phases. Prototyping is one of the easy solutions for defining and refining requirements. After gathering the user stories, the developers may prepare a prototype. This prototype includes the functional and non-functional needs of the user, which are produced from user stories. However, these specified needs may be false, because the user may explain the needs false or the user may understand his needs realistically. When the users and developers work on a prototype, they can agree on real needs of the software project. After two sides of the project agree on the prototype, the coding phase will start. While this operation, the documenter should prepare two documents for a better work: The project dictionary is the first document and should be updated to know the terminology of the project. The General Specification document is the secondary result of this step. This document includes the boundaries and validation criteria of the software project. For each module, the collection of user stories phase should be reported with a detailed specification document. This document should report the work on elicitation process and specify also the details of the module according to the user needs.

- *Coding – Unit tests:* The coding phase is the easiest phase step of "Rings" to define. The simple programming of the software is the meaning of this step. In this phase, pair programming may be found useful and may be used. But, the human resource of the project is significant. If the human resource is enough, pair programming is preferred. While coding, the unit tests will be also applied to the inner units (Or functions) of the system to verify it. Each coding step should be finished with this unit tests and the results of these test should be reported via the documenter.

- *Integration – Feedbacks:* Continuous integration is a meaningful practice. To specify the problems of product, the errors of the communication between the units is as important as the errors of the units. So the produced units should be integrated as often as possible. To validate the project, the feedbacks are too important. “RINGS” accepts that the most important information source of a software project is the user of the product. So, each version will be delivered to user to take feedbacks about the verification and validation of the project. According to the feedbacks, the development process may return to the coding phase to correct the faults or the process may go with another module and return to a new “user stories” phase. After the acceptance, the documenter will update the documents for users or future products; procedure documents and user help documents.

Our model doesn’t advice any process model for a company which is fully adapted to the 3rd circle, because the company in 3rd level have to have someone to check the current process cycle and new technologies like Software Engineering Process Group. This group should revise the current process cycle according to the new technologies, project types, company ethics, etc. As a result, the 4th and 5th circles will be drawn differently for each company.

2.3.3 Team Requirement of RINGS (Roles of Team Members)

“Rings” is a complete system with detailed information for a small company. This information includes not only the process steps, but also the role definitions of the development team.

At the first ring, the only coordinator of the system is the project manager. The project manager tries to coordinate all of the group members and development process. It is expected that there is a company hero in the development process. This

hero has a good experience and knowledge about software development. However, he doesn't have any experience and request to be the manager of the process. Because of this, he can't be the manager of the system. The project manager specifies the modules, prepares the version list with help of hero and traces the process according to the foresights.

In the middle ring, the management group includes 3 people and a project management tool. The management tool will help the managers by defining the tasks and tracing them, because the development group is not large enough to use a managers group. With the project manager from the 1st circle, we will take the development hero into the management group as surgeon, because we expect that the hero has now some experiences about management. Another reason is the increasing project number. The company may need several heroes in several projects. So if the hero doesn't belong only to one project, it will be cheaper. The surgery doesn't have a specific task from modules to do. However, he should be in all of the modules of the software. The surgery will attend to all of the module developments when needed like a technical manager and solves general problems like database or manner changes. The last member of this 4 people management group is a documenter. The documenter will try to report all of the process and development to prepare the documentation. So the documentation phase may be ignored from here.

The management group in outer circle increases by 1 person from the previous circle. A quality manager will be added to management group. The quality manager will check the system and improve the quality standards. The whole management group works also as a Software Engineering Process Group. The whole group represents the development group in the verification and validation meetings.

The role definitions are explained according to the ring names. When the roles are focused on, the duties, responsibilities and skills may be explained as follows;

2.3.3.1 Developers

The developers are simply the coders of the projects. The developer's role has defined in first ring and continues in other rings. They are the backbones of the software project. The "Continuous Coding" value of "RINGS" puts the importance forward.

According to the continuous coding value, the first and most important duty of a developer is preparing the product. The preparation term begins with the requirements analysis of the modules (/tasks). The developer will attend the meeting for user stories in 1st ring, but in 2nd and 3rd rings, the developers don't attend the meetings, in which the project basics will be specified. After the project basics are determined, the user stories of each task (/module) will be also gathered by the developers. Afterwards gathering the user stories, the developer will prepare the automated tests and code the specified requirements. The verified code will be integrated to the system and validated by the user. This feedback phase will be also checked by the developer. The developer will note the output of the feedbacks and correct the problems. After the prepared task is finished and integrated into the system without a problem, the work for this module is finished and the developer may go to the next module (/task/component/etc.)

The developers should work as pairs, if possible. The Pair Programming practice of Extreme Programming is very useful and the advantages of the pair programming should be taken. The crew may be not enough in 1st ring, but in 2nd and 3rd rings, the teams will be big enough to form pairs for development. The first advantage of pair programming is the reduction of risk of loosing some members from team. When two members of team work on a module, the risk will half. Additionally, the better performance on development will be one more advantage.

The most important responsibility of a developer is coding the requirements and producing the software. So, the developer will be responsible of documentation only in 1st ring. This documentation is also defined as a separate phase. The developer

should focus on continuous coding and after the product is prepared, the documents will be written by the developers. In 2nd and 3rd rings, the documentation duty will be left to another role.

The developers should be experienced on coding. Because the system forces the developer to really develop (not implement!) the software, the tricks and tips about the programming language and Integrated Development Environment (IDE). So, he can easily focus on analysis and design phases of a development. Additionally, the developers should be good team members. They should work with other team members without having trouble. The oral communication and analysis skills of a developer are also very important, because he should do his own requirements analysis and the most important information transmit method will be the oral discussions in the team.

2.3.3.2 Project Manager

Project Manager is coordinator of the project. The Project Manager role will be chosen in 1st ring. The Project Manager of the 1st ring will be the overall coordinator of the team and will help to team for development, if needed. The first and most important duty of a manager is keeping the communication channels of the team open. The communication channels between the development team members and between the developers and customers are important. So, the manager should solve conflicts, arrange and administer meetings and be the interface between customers and developers. Secondary duty of a project manager is planning the project and following the success of the plan. The plan should be realistic and may contain not detailed estimations about the inner process of the functionalities. In 1st ring, detailing the functionalities may be the work of the project manager, but in 2nd and 3rd rings, the surgeon will help him.

In 1st ring, the project manager will be responsible for preparing and following the version plan, solving conflicts in the system, coordinating the team and reporting the system online. If needed, the manager may also help coding. In 2nd ring, the

responsibilities of Project Manager will reduce. The Manager will not code and write documents. But because the team and the project enlarged, the manager's problems will also enlarge. And in 3rd ring, the group that the manager will coordinate will increase. The user stories to manage and the modules to implement will also enlarge. Thus the project manager should be more focused on developers, customers and interactions between them.

The project manager can make all of the decisions, which will affect the project, except the unsuccessfully cancellation decision.

The project manager should be a good leader and coordinator. The team will follow the leader of the project, when there are problems with software and with customers. So the leadership and communication skills are very important.

2.3.3.3 Documenter

The documenter will be added to the project management group in 2nd ring. The documenter will be responsible for producing the documents. Because of the "Continuous Coding" rule, the developers don't prepare documents. Their only duty is developing software. So, the documenter will follow the development cycle with all of the members of team and fill in the blanks in the documents. The documenter should follow the developers and prepare the documents according to the codes and coding phase. Also, the documenter should follow the project manager and other management roles to document the plans and management decisions.

The documenter doesn't have any right about deciding something in project.

The documentation is not an easier work in project than developing or managing it. The documenter should follow all of the information transfers and transactions in the system to write down. So, the communication and intelligence skills are very important for a documenter. Additionally, the documenter should be systematic to handle too many information at the same time. The handled information should be

documented regularly and so, the documenter has to explain other's information well to other people, because he should be a transmitter (or "Under-liner") between the source (programmer or customer) and the target (customer or programmer). Because of this, the documenter should have a good skill of writing reports. The biggest problem for programmers is preparing documents. But there can not be used a standard secretary to prepare the reports, because the reports will contain information about software engineering and development. The member, who will document someone's code, should have information about the process that they use and about coding the software. As result, the documenter should be a Computer Engineer or an experienced coder.

(Additional (Practical) information: The required skills for a documenter are too much as seen. The most important and most problematic skills are having domain information and being regular documenters. The software engineers don't like documenting and the regular documenters don't know the secrets behind software development (either theoretical or practical secrets) Therefore, finding a documenter will be a nightmare. As solution, it can be advised to hire interns or part-time workers from Computer Engineering (or related departments) students. It will be useful, because these workers will be hired with this work information and can't refuse. Another advantage is their information about domain. These workers will have primitive software development knowledge, and this will be enough to follow the system and prepare documents. (Regular documenting skill will be trained.) The last advantage of such a work will be the adaptation and learning process of the worker. The possible workers of future may be hired in this way and they may be adapted to the work by not giving too many responsibilities and by taking him into the process at every step.)

2.3.3.4 Surgeon

The project hero is one of the most effective and most problematic members of a software development team. Project hero is the well-informed and most creative developer in the team. He may be also well-experienced and have good information

about domain. However, the project hero is most problematic member, because their common property is also non-manage-ability. The project hero may solve all of the problems, but may not work under straight rules and with pairs. While forcing him for such works, the performance of entire team may decrease. The surgeon is a more managerial role for a project hero. The purpose of this role is using the creativity and domain information of the hero, but avoiding him to work with pairs and decreasing the communication with other members.

The surgeon will create the system design and prepare the task lists with the other members of the team. Because of these duties, the surgeon may be also called as a technical leader of the team. Additionally, the surgeon will prepare the adaptations of new technologies into the projects. With this work, the surgeon may be called as an instructor in the team. As a third work, the surgeon will be work as a developer, if needed. If the developers need, they can call the surgeon to solve an algorithmic, a technological or a design problem for a short time. At this time, the surgeon can use his lancet for a short time and leave the whole designing, coding and testing work to the developers according to his solution.

The surgeon should be in decision meetings for first project analysis and version planning with the customer. The technical requirements and information from experience will be added to the project via this way. The project manager will decide everything, but the surgeon will assist him, while making technical decisions.

The surgeon will have the similar skills as a programming hero with his negative and positive skills. As positively, a surgeon should be well-experienced and creative. The technical background and the competence of capability will be also important while choosing a surgeon. If a team has several project heroes in 1st ring, some additional skills may be thought for choosing the surgeon from them, because more than one surgeon will be too much for a team which is not bigger than 15-20 members. Additionally, a better communication skill, pair working ability, teaching-capability and harmony with project manager may be useful as criteria.

2.3.3.5 Software Management Tool (Software)

The management of a team is always a problem. In 1st ring, the project manager may manage the project. However, after 2nd ring, the team and size of the project will be enlarged. To keep the system updated and controlled, someone should keep everything in mind and force to remember the plan during making decisions.

A software management tool may gather information for decisions or may be programmed for making decisions automatic.

This tool will be used to process something automatic. So, the first skill will be keeping the team tasks and tracking the system according to this plan. Preparing detailed automatic reports from the submitted data will be another useful skill of the tool. The automated reports will ease the work of team for reporting the development process for executives or standard checkers.

2.3.3.6 Quality Manager

Quality manager may be called as overall controller of the project. The first duty of the quality manager is checking the works of project manager, surgeon, documenter and developers. The project manager's responsibilities are defined and the quality manager checks the minimum requirements and quality of work. The developer's programming/developing work will have some coding standards and working standards. Quality Manager will check these standards of developers. The prepared standard metrics for software development process may be used for specifying the quality of the work mathematically. Another work of Quality Manager is preparing the standards and applying new process ideas in the company.

Quality Manager is responsible for designing overall process for products in a good quality. For this reason, he may be seen more qualified than the project manager. In the company, this answer may be true. However, the project manager is authorized on project for the vision of executives and customers.

Quality Manager should be the leader of the project. The developers should see him also in the same authorized level with project manager. For having an effect like this, the quality manager should have a good leadership and communication skill. One of the most important skills of a good Quality Manager is the understanding skill. The quality manager should understand new technologies and new software engineering trends. Other important skills for Quality Manager will be also being attentive and being regular. The quality manager should gather information from several ways from the members of team. This information should be well arranged and documented for a right decision on the quality. Additionally, a more attentive manager may easily gather the required information.

2.3.4 Documentation Requirement of RINGS

Preparing documents is the process, which differentiate software from a simple program. The advantages of preparing documents are too many. Some of them;

- Preparing documents will make software visible for customers
- The documents will make the requirements visible for developers
- The documents will make the software discuss-able for developers
- The documents will make entire information accessible
- The documents may be used for a better/faster maintenance
- …

As seen, documenting a process and a product is useful. However, it is not so easy to prepare the documents. Because the developers are focused on developing solution, they can't find enough time for documentation and writing documents are not in skill checklist for choosing the developers. As result, the developers don't like preparing documents. By accepting this problem, "RINGS" will advice "Continuous Coding" principle. "RINGS" don't want other works rather than coding for developers. Several solutions for this problem is put forward by the new

methodology like a separate “Documenting” phase step and a separate “Documenter” role in the team.

In the first ring, the user stories should be documented after the first requirements engineering phase. There is a specification report before version planning. After version planning the version plan will be published. In the documentation phase in the first ring, the general documentation should be done. (For example; procedure information, user manuals …)

From inner ring to middle ring, some standards should be developed. A coding standard for the company has to be prepared according to the needs and experiences of the company.

In the middle ring, the boundaries of the software will be prepared from the first requirements engineering phase. The secondary document from this phase is also the quality standards for the project. Like first ring, a version planning document will be published.

From middle ring to outer ring, a documentation standard and the general quality standards document will be published.

In the outer ring, an archive document will be published after each phase. These documents will be traced and checked by customer with a feedback. These documents include; project boundaries, resources, version plan, database – user interface basics, procedure library, resource updates, … Some additional documents like a project dictionary should also be added to the project. This dictionary will contain the meanings of the technical terms in the concept of the software.

Documentation is the most important thing for management and coordination. For a better management and easier maintenance, required things should be well-documented. The requirement of documentation increases ring-by-ring. The documents of “RINGS” are as follows;

2.3.4.1 General Specification Document

General Specification Document will be used in 1st, 2nd and 3rd rings. In 1st ring, the General Specification Document is the first and only document to report the requirements and needs of the user during the development. After development of the product, some detailed requirements documents may be produced according to the customer requests. In 2nd and 3rd rings, the document contains the information, which is gathered from "Project Basics" step. So, the document has only the basic needs and most important constraints about the project. The inner details of modules (or sub-projects) will be detailed in the document, which will be prepared after "User Stories" phase.

The report will be prepared in a formal way. The language should be formal (not native), but it should not be technical. The target readers of this document will be users, customers, developers and executives. As result, the document should be readable for all of these classes. A small General Specification Document should at least contain the following sub-chapters:

- General description of the product
- Schedule constraints from User/Customer
- Cost estimations
- Technical constraints
- Gathered User Stories
    - Description
    - Priority
    - Accepter (if possible)

The General Specification Document is the responsibility of the management group of the team. In 1st ring, the team can't be divided and the writer group will be all of the team. (The Project Manager will be actual writer.) In 2nd and 3rd ring, the

management group will attend to the “Project Basics” meeting and prepare the document. The documenter of the management group will be the actual writer of it.

2.3.4.2 Version Plan

Version Plan will be produced on each ring after version planning phases. The plan may be seen the obligation of the development group to the customer about the delivery. The versions will be prepared by the management group and contain the functionalities of each delivery.

Version Plan has not so different and complex structure. So, a structured report may be enough and useful. The structure may contain the following information at least;

- General Version List
- Due dates for each version
- User stories for each version
- Accepter for each version

The development group should prepare the version plan. After preparation this plan, the project manager may write down the document. In 2nd and 3rd rings, this document will be written by the documenter member of team.

2.3.4.3 User Interface – Database Design Document

User Interface and Database are two base things for a software project. The designs of these bases should be documented to memorize the desires for critical decisions, either by the users or by the developers. This document will be prepared in all of the rings. In 1st ring, only one document may be enough. However, in 2nd or 3rd rings, the document should be updated after each module development.

The language of the report should be as structured as possible. Because the report will be a help for developers, the structured plan will be useful for a technical and focused reading. The structure may be as follows;

- General User Interface description
- Interface prototypes for each module
- Interface prototypes for critical functions
- Reasons for interface decisions
- General Database description
- Database / Table structures
- Reasons for database decisions

This document will be produced by the developers in 1st ring. In 2nd and 3rd rings, the documenter will write down the document after the meetings of the management group. The updates on the User Interface/Database Design document will be also done by the documenter.

2.3.4.4 Procedure Documentation

Procedure documentation contains all of the documents about the procedures. The reason of such a document is to maximize the reusability of the program. The details of the procedures will be prepared at the end of the project in 1st ring. At 1st ring, this document will be written at the last phase of the process. At 2nd and 3rd ring, this document will be prepared after each integration phase.

The procedure documentation will be prepared as a structured manner. The documentation may be thought in two parts. First part of the procedure documentation is re-writing the codes according to the standards and commenting. Second part is a separate document, which contains the following information;

- Procedure Name
- Procedure Description/Definition

- Procedure Inputs/Outputs

The documentation will be prepared by the developers at first ring. However, at 2nd and 3rd rings, the responsibility will be taken by the documenter. The documenter will write the document according to the information from the developers. The code commenting should be also done by the documenter. But the developers may be often needed, if the coding standards are not so descriptive.

2.3.4.5 User Manuals/Helps

User Manuals and Helps will be prepared at “Documentation” phase. However, at 2nd and 3rd rings, the documenter should also prepare the manuals, while the project goes. In the first phases, the documenter may prepare the overall structure of the manual and in the following phases, the structure may be filled.

The manuals and helps are product-oriented and customer-oriented reports. So, not a common report format may be prepared for the manuals. However, according to the reason of the document, it can be said that the document will contain usages of the methods, critical use-cases and definition of the user interface.

The document should be prepared by the developers at 1st ring. However, at the other rings, the documenter will prepare the documents and approve by the customers.

2.3.4.6 Coding Standards

Coding Standards document is a standardization effort of the team to maximize the code ownership and pair programming. The document should be prepared before running the 2nd ring according to the addictions of the team and the successful examples from the market.

The document should be prepared in a structured manner, because it will be used as a reference manual. The standards will be;

- General description
- Overall structure for a program
- Overall structure for a module
- Commenting syntax and semantic
- Naming standards
- Abbreviation standards
- Test Code standards
- Test Data standards

The document will be prepared and used by entire team.

2.3.4.7 Specification Document

Specification Document is a minimal type of General Specification document. It will be started to be prepared from 2nd ring. The document will contain detailed information about the modules (or sub-projects).

Because the document will be read by the user, customer and developers, the document should be prepared for different types of readers. The document should contain besides some information, maybe in native language, for users and customers and also some structured information for developers and maintainers. At least, the following sections should be in document;

- The cross-reference to general specification document
- New user stories
- Functional and non-functional requirements of the module
- Task specifications for the module
- Versions of this document

The specifications about the modules will be done by the project manager and surgeon of the project. The documentation of these meetings will be done by the documenter and should be approved by the attendees of the meetings. (Project manager, documenter, customer representatives, and users)

2.3.4.8 Unit Test Document

Because “RINGS” accepts the Extreme Programming, automatic unit testing is one of the important things of coding phase. In 1st ring, the developers will also use test-first design manner, but they should not document the unit tests, because the developers don’t have so many time resources and it is not necessary to document the unit tests for a project of 1-3 months. However, in 2nd and 3rd rings, the growing team and growing project resources will force the team to document and archive the unit tests.

The unit tests will be in a structured manner, because the first resource of this information will be the automatic testing tools. And the information from these tools should be documented for easier archiving. The structure of this document;

- The general description of the module to be tested
- Name of the unit test
- Reason of the unit test
- User Story of the unit test (if possible)
- Code of the unit test
- Data of the unit test
- Commenting standard of results of the unit test

This document will be prepared by the documenter according to the information from automatic tools and developers.

2.3.4.9 Documentation Standards

The documentation standards will be prepared before the team goes to the 3rd ring. Before this ring, the documents may be prepared according to the standards of "RINGS" and the standards of the literature. But, after 2nd ring, the development group should produce their own standards for documentation.

The documentation standards document is a structure for other documents. So, the document will contain all of the future documents and detailed information for them.

The standards will be prepared by the project manager, documenter and new-coming quality manager. The quality manager will write down the standard rules besides for the format and also for the content of the documents of the system.

2.3.4.10 Quality Standards

Quality standard is a new term for software development. In 3rd ring, the team should have standard quality metrics to measure. So, in 3rd ring, the written quality standards will be used to define the process.

The quality standards will have two types of information. Overall quality standards for the process and company may be defined in native language. But the standards for program codes and documentations should be automatic measurable and structured. As example for measurable standards;

- Standard structures for documents
- Development speed metrics
- User story standards
- Commenting standards

The quality standards may be used in all of the rings. But, the standards will be first documented in 3rd ring by the quality manager and project manager.

2.3.4.11 Schedule Document

The schedule document is an approval and discussion document. In 1st and 2nd rings, the schedule document may be not necessary, because the system has not so much time resources and the team has not so much human resources for preparing so detailed reports.

The document will have not so different information. Because of this, the structure is not so large;

- General iteration plan
- Name of iteration
- Description of iteration
- User stories for the iteration
- Deadline for the iteration
- Possible team arrangement for the iteration

The document will be prepared by the management group. The project manager and surgeon will be first responsible management group members by preparing the document. The Quality Manager will review the documents and discuss the quality and feasibility of the document. The documenter will write the all project and team memory.

2.3.4.12 Cost Document

The cost document is another version of schedule document and is also an approval and discussion document. In 1st and 2nd rings, the cost document may be not necessary, because the system has not so much time resources and the team has not so much human resources for preparing so detailed reports like schedule document.

The document will have not so different information. Because of this, the structure is not so large;

- General iteration plan
- Name of iteration
- Possible cost of iteration
- Possible financial resource for the iteration

The document will be prepared by the management group. The project manager and surgeon will be first responsible management group members by preparing the document. The Quality Manager will review the documents and discuss the quality and feasibility of the document. The documenter will write the all project and team memory.

2.3.4.13 Pricing Document

The pricing document will be produced according to many different parameters. The cost document will be the base of this document. This document will be prepared in the 3rd ring.

The language of the pricing document should be also divided into two parts. The native language will be used to define the reasons of the pricing. After that, a structured table will be used to detail the prices.

The members, who will prepare the document is so many like the parameters. The pricing document will be prepared by the project manager and surgeon, but be reviewed by the executive.

2.3.4.14 General Module Specification Document

General Module Specification document is a result of the estimation phases. The document is prepared after version planning phase according to the information from the schedule and cost documents in 3rd ring.

The specification document is not so different then other specification documents in the same ring or previous rings. Because the document will be read by the user, customer and developers, the document should be prepared for different types of readers. The document should contain besides some information, maybe in native language, for users and customers and also some structured information for developers and maintainers. At least, the following sections should be in document;

- The cross-reference to general specification document
- New user stories
- Functional and non-functional requirements of the module
- Task specifications for the module
- Versions of this document

The specifications about the modules will be done by the project manager and surgeon of the project. The documentation of these meetings will be done by the documenter and should be approved by the attendees of the meetings. (Project manager, documenter, customer representatives, and users)

2.3.4.15 Project Dictionary

Project dictionary is the naming strategy of overall system. Each term of the system has two meanings. One of them is the meaning of the term for the user and the other is the meaning for the developer. To define and standardize the terminology for the system, the project dictionary will be needed in 3rd ring.

The project dictionary should be in a structure. The columns of a structure may contain the following information;

- The term
- The description of the term for user
- The description of the term for developer
- The coding type of the term (if possible)

The project dictionary will be prepared by the developers, surgeon and users. After that, the documenter will document and update the list.

2.3.5 Advantages

Software Development Methodologies are only thinking according to the properties of the current company. However, “Rings” starts with a young software company and tries also to design the future of it. This is the time parameter of “Rings”. It brings the time parameter into the software development processes. So, the software development process of the company begins with a simple form and grows up into a more complex form according to the augmentation of the problems and resources. This situation provides also easier adaptation to disciplined methodologies.

The parts of “Rings” are familiar to the software development companies. The theme that the method focuses on is the changing requirements and resources of the company and the obligation of process changes. So, “Rings” focuses on the resources of the companies and try to prepare a roadmap to the young company to be a mature one.

Each phase of the “Rings” is developed according to the incremental development manner and agile software development ideas. So the rapid development and higher satisfaction of customer are the expected advantages of it, because the base methodologies and ideas are the proofs for these results.

2.3.6 Risks

“Rings” is not already widely used. So, the usage problems and advantages are not proofed by the companies. The advantages are only expectations because of their base methodologies. The widely used base ideas are the guarantee of success of “Rings”.

“Rings” is prepared for a young company. The expectations for this company are; low resources, chaotic programming, low discipline, low documentation and small projects. The method starts in this case and provides an incremental expansion. However, if the starter company tries to break the “Rings” from a ring and start to use it from there? This situation is a risk. The system is solid with its process steps, principles and implementation. So, when the company breaks the implementation order of “Rings”, this may break all construction of the system.

One of the most important disadvantages of agile software development methodologies is the dependency to the human-being. “Rings” is dependent to the work of the developers and managers. The risks (especially in first ring) cannot be handled with high discipline and documentation, because of the high agility and this human-dependency may cause some problems in application according to the experience of the developers.

CHAPTER THREE
APPROVAL OF RINGS

3.1 Overview

Overall software development methodologies and “Rings” is defined in previous chapters. The most critical risk of the new methodology is the risk, that the method is not used in any company. Therefore, it will not be seen trustworthy.

Two solutions may be found to build trust;

- Checking the maturity of the methodology (the company, who ideally uses “Rings”)
- Taking comments of the possible users of software market

The first approval method is for checking the method for the general rules of software market. Capability Maturity Model is prepared with the comments from different parts of the software market. The needs of customers, developers, executives, etc. had collected to guarantee the success, before starting the project. The model works with a checklist for the ideal company (mature company). As an approval model, the ideal application of “Rings” may be checked according to this checklist. When all (most) of the entries in the checklist can be covered with the rules of “Rings”, then it can be said that the company, who applies all rules will have the defined maturity level.

Comments of the professionals may be used as a secondary approval of “Rings”. The method will be used in the market and therefore the comments of them will be very important. For such an approval, a questionnaire was prepared for taking the existing problems, which may be solved with changes in development method. Additionally, interviews were made with these professionals about the usability of “Rings”.

3.2 Formal Approval

3.2.1 Capability Maturity Model

One of the most important problems for customers is the reliability of the developers. To show the reliability of a developer, the Carnegie Mellon University Software Engineering Institute has developed one of the famous accreditation models. By using this model, the customers may see how the developer is reliable. (The reliability here can be defined as conforming to time/cost specifications, obtaining high maintainability, expandability, etc.)

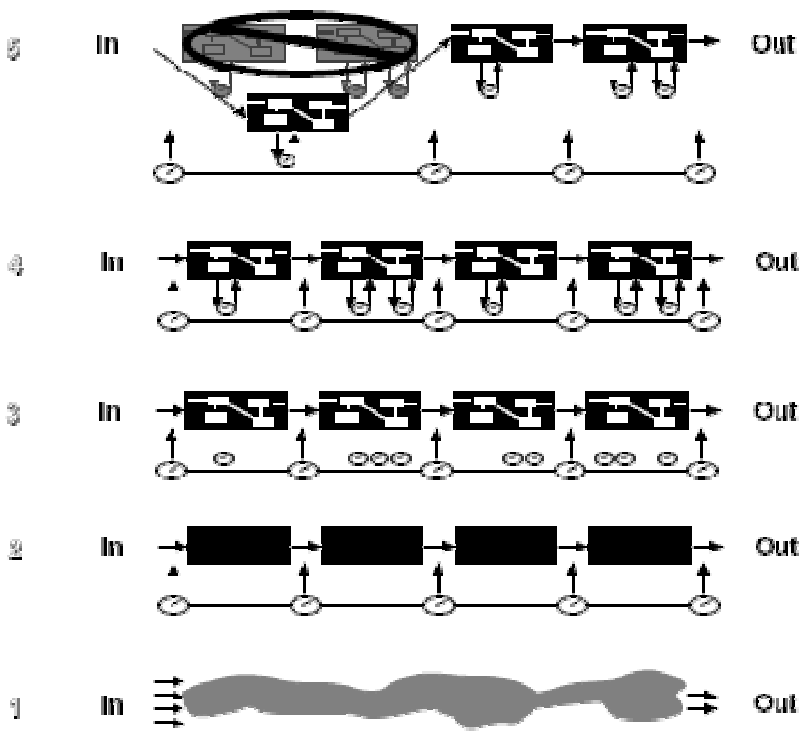

Figure 3.1 Process stages of Capability Maturity Model

This famous accreditation system is called CMM (Capability Maturity Model). Capability Maturity Model is a model to classify the software development companies. This model has 5 stages for different levels of maturity. (Paulk, Curtis, Chrissis and Weber, 1993)

Initial stage: This Maturity Model stage is a heroic system. There is a software hero, who can solve all problems. The process is not defined. The results may be successful, but it is not guaranteed, because the overall system is human-oriented. There is no defined checklist for this maturity level, because this level determines that nothing about software engineering can be seen in the company.

Repeatable stage: In second stage, the process is defined to repeat the process of a successful product development. The developers are not experienced with software engineering practices, but they have capabilities to implement previously tried process. For approving a company as mature at Level 2, the following checklist should be covered:

Requirements Management

Goal 1: System requirements allocated to software are controlled to establish a baseline for software engineering and management use.

Goal 2: Software plans, products, and activities are kept consistent with the system requirements allocated to software.

Software Project Planning

Goal 1: Software estimates are documented for use in planning and tracking the software project.

Goal 2: Software project activities and commitments are planned and documented.

Goal 3: Affected groups and individuals agree to their commitments related to the software project.

Software Project Tracking and Oversight

Goal 1: Actual results and performances are tracked against the software plans.

Goal 2: Corrective actions are taken and managed to closure when actual results and performance deviate significantly from the software plans.

Goal 3: Changes to software commitments are agreed to by the affected groups and individuals.

Software Subcontract Management

Goal 1: The prime contractor selects qualified software subcontractors.

Goal 2: The prime contractor and the software subcontractor agree to their commitments with each other.

Goal 3: The prime contractor and the software subcontractor maintain ongoing communications.

Goal 4: The prime contractor tracks the software subcontractor's actual results and performance against his/her commitments.

Software Quality Assurance

Goal 1: Software quality assurance activities are planned.

Goal 2: Adherence of software products and activities to the applicable standards, procedures, and requirements is verified objectively.

Goal 3: Affected groups and individuals are informed of software quality assurance activities and results.

Goal 4: Noncompliance issues that cannot be resolved within the software project are addressed by senior management.

Software Configuration Management

Goal 1: Software configuration management activities are planned.

Goal 2: Selected software work products are identified, controlled, and available.

Goal 3: Changes to identified software work products are controlled.

Goal 4: Affected groups and individuals are informed of the status and content of software baselines.

Defined stage: There is a standard process definition. This definition can be re-defined according to the small changes or software product types. The process definitions are defined for management and development phases. The developer team is also well-informed about software engineering discipline and processes. There is also a Software Management Group to check the on-going process and measure all phases of development according to the defined metrics. The checklist for the Level 3 is following the section;

Organization Process Focus

Goal 1: Software process development and improvement activities are coordinated across the organization.

Goal 2: The strengths and weaknesses of the software processes used are identified relative to a process standard.

Goal 3: Organization-level process development and improvement activities are planned.

Organization Process Definition

Goal 1: A standard software process for the organization is developed and maintained.

Goal 2: Information related to the use of the organization's standard software process by the software projects is collected, reviewed, and made available.

Training Program

Goal 1: Training activities are planned.

Goal 2: Training for developing the skills and knowledge needed to perform software management and technical roles is provided.

Goal 3: Individuals in the software engineering group and software-related groups receive the training necessary to perform their roles.

Integrated Software Management

Goal 1: The project's defined software process is a tailored version of the organization's standard software process.

Goal 2: The project is planned and managed according to the project's defined software process.

Software Product Engineering

Goal 1: The software engineering tasks are defined, integrated, and consistently performed to produce the software.

Goal 2: Software work products are kept consistent with each other.

Intergroup Coordination

Goal 1: The customer's requirements are agreed to by all affected groups.

Goal 2: The commitments between the engineering groups are agreed to by the affected groups.

Goal 3: The engineering groups identify, track, and resolve intergroup issues.

Peer Reviews

Goal 1: Peer review activities are planned.

Goal 2: Defects in the software work products are identified and removed.

Managed stage: The development group should conform the three phases of Maturity model. Additionally, the defined metrics are tried to measure and comment and this will obtain to control the production of the development group. The development group can make better foresights by using the collected results from the previously implemented projects. The addition to the checklist for Managed stage is as follows;

Quantitative Process Management

Goal 1: The quantitative process management activities are planned.

Goal 2: The process performance of the project's defined software process is controlled quantitatively.

Goal 3: The process capability of the organization's standard software process is known in quantitative terms.

Software Quality Management

Goal 1: The project's software quality management activities are planned.

Goal 2: Measurable goals for software product quality and their priorities are defined.

Goal 3: Actual progress toward achieving the quality goals for the software products is quantified and managed.

Optimized stage: The software developer group has a defined process. The developers have a background about the software engineering. The Software Engineering Process Group controls the process to improve the production with the current definitions. However, the companies on 5th level can optimize their process model according to the new technologies and process models.

Defect Prevention

Goal 1: Defect prevention activities are planned.

Goal 2: Common causes of defects are sought out and identified.

Goal 3: Common causes of defects are prioritized and systematically eliminated.

Technology Change Management

Goal 1: Incorporation of technology changes is planned.

Goal 2: New technologies are evaluated to determine their effect on quality and productivity.

Goal 3: Appropriate new technologies are transferred into normal practice across the organization.

Process Change Management

Goal 1: Continuous process improvement is planned.

Goal 2: Participation in the organization's software process improvement activities is organization wide.

Goal 3: The organization's standard software process and the projects' defined software processes are improved continuously.

3.2.2 Results of Capability Maturity Model

The “Rings” process model cannot be accredited unless it is used widely in several companies. To accredit the model, we should observe at least 3 companies and at least 5-7 projects (approx. 10-15 years). Because it is practically too difficult to observe, the goals of the CMM are tried to check. It is also checked by the responsibilities of the team members or document requirements, whether the goals will be satisfied or not.

For Level 2: “Repeatable”;

Requirements Management

Goal 1: System requirements allocated to software are controlled to establish a baseline for software engineering and management use.

Solution in “Rings”: The Project Manager will arrange the resource allocation and define the project basics at the first step. According to the

size of the project, there is no need for a detailed resource management. (The suggestions are about small or medium size projects.)

Goal 2: Software plans, products, and activities are kept consistent with the system requirements allocated to software.

Solution in "Rings": Project Manager will be responsible to define these plans. Documenter and Project Management Tool will help him to keep the plans, products and activities documented and consistent.

Software Project Planning

Goal 1: Software estimates are documented for use in planning and tracking the software project.

Solution in "Rings": The Project Manager and Surgeon will estimate the allocations according to the management and technical thoughts. Documenter will document these estimations and updates them when required.

Goal 2: Software project activities and commitments are planned and documented.

Solution in "Rings": The activities will be defined after Project Basics and "Version Planning" phases by Project Manager and Surgeon. The General Specification Document and Version Plan Document contain the activities and commitments. Documenter prepares these documents and updates when required.

Goal 3: Affected groups and individuals agree to their commitments related to the software project.

Solution in "Rings": The outputs of the first two phases, Specification Document and Version Plan will be signed by the developers and customers.

Software Project Tracking and Oversight

Goal 1: Actual results and performances are tracked against the software plans.

Solution in "Rings": The Project Management Tool will track the project according to the inputs from the decisions of Project Manager and Surgeon. Documenter and Project Management Tool will document the outputs.

Goal 2: Corrective actions are taken and managed to closure when actual results and performance deviate significantly from the software plans.

Solution in "Rings": There is a break after each module in the feedback phase. At this break the project manager and surgeon may think over the plans. The experienced Project Manager and experienced Surgeon may re-arrange the prepared development plans.

Goal 3: Changes to software commitments are agreed to by the affected groups and individuals.

Solution in "Rings": Some changes may be done in the feedback phase. These changes will be applied to the documents. The new plans will be also documented by the Documenter and the Documenter is also responsible by confirming the new documents.

Software Subcontract Management

Goal 1: The prime contractor selects qualified software subcontractors.

Solution in "Rings": Because the thought software development projects are not so large, the company will not need to subcontract management.

Goal 2: The prime contractor and the software subcontractor agree to their commitments to each other.

Solution in "Rings": Because the thought software development projects are not so large, the company will not need to subcontract management.

Goal 3: The prime contractor and the software subcontractor maintain ongoing communications.

Solution in "Rings": Because the thought software development projects are not so large, the company will not need to subcontract management.

Goal 4: The prime contractor tracks the software subcontractor's actual results and performance against its commitments.

Solution in "Rings": Because the thought software development projects are not so large, the company will not need to subcontract management.

Software Quality Assurance

Goal 1: Software quality assurance activities are planned.

Solution in "Rings": The quality issues may be divided into two sections: The quality of the software as the functional quality of the product and the process quality of the company as the non-functional quality. The non-functional quality issues will be planned by Project Manager. And the functional quality issues will be planned by Surgeon. These two plans will be defined at "Project Basics" level.

Goal 2: Adherence of software products and activities to the applicable standards, procedures, and requirements is verified objectively.

Solution in "Rings": For an objective verification, the project management tool was added to the system. The verification of the standards will be applied to this tool and the tool will check the process. (The responsibilities of the Project Management Tool were defined at previous sections.)

Goal 3: Affected groups and individuals are informed of software quality assurance activities and results.

Solution in "Rings": There is a documented coding standard in the system. This standard will ensure mostly the product quality. For other quality assurance activities the project manager will be responsible.

Additionally, the plans and successful results will be documented by Documenter and he is also responsible to let the customers check the plans.

Goal 4: Noncompliance issues that cannot be resolved within the software project are addressed by senior management.

Solution in "Rings": There can be always problems, because there are two domains in the system. First of all, the "Rings" is built on the better direct communication in the affected groups. This mental work will reduce the noncompliance issues. However, when some problems occur in the project, these problems will be addressed to the managers in the system according to the "Coordination" principle. Briefly, for managerial problems, the Project Manager is responsible to solve them. And also Surgeon is there to solve technical problems.

Software Configuration Management

Goal 1: Software configuration management activities are planned.

Solution in "Rings": Software Configuration Management is thought over version planning. Versions will be planned detailed in the "Version Planning" phase and tracked by the Project Management Tool and Project Manager.

Goal 2: Selected software work products are identified, controlled, and available.

Solution in "Rings": The general identification and control of the work products will be handled by the Project Manager and Surgeon in the "Project Basics" phase. But, the detailed identification of the work products will be done by the developers in "User Stories" phase for each module. The control and availability checks are thought over "Configuration Management" of the software. Configuration Management is one of the features of Project Management Tool. This tool and surgeon is the controllers of the system. (Tool will be used for a live control and objective

verification. Surgeon will be used for technical and inner control of the work products.)

Goal 3: Changes to identified software work products are controlled.

Solution in "Rings": All changes are handled by the documenter for updates in the prepared documents and Project Management Tool for archiving and online controls. In short, the identifications will be controlled by the Documenter and the Project Management Tool.

Goal 4: Affected groups and individuals are informed of the status and content of software baselines.

Solution in "Rings": Documenter will document all of the configuration management activities and is responsible to be checked by the affected groups. (Additionally, all of the baselines are open to entire group members with the Project Management Tool.)

For Level 3: "Defined"

Organization Process Focus

Goal 1: Software process development and improvement activities are coordinated across the organization.

Solution in "Rings": The Project Manager will have entire coordination responsibility in team. The development process is overall documented in "Rings". The Rings methodology is a complete methodology with its process, team members, documentation requirements and process improvement cycle.

Goal 2: The strengths and weaknesses of the software processes used are identified relative to a process standard.

Solution in "Rings": The Quality Manager is the only responsible member of the team about the process improvement of the company. However, "Rings" is prepared as a complete set with its improvements. So,

when the weaknesses of the current ring increase according to the project size, team size, etc., the ring will change.

Goal 3: Organization-level process development and improvement activities are planned.

Solution in "Rings": Nothing about the organization-level can be determined.

Organization Process Definition

Goal 1: A standard software process for the organization is developed and maintained.

Solution in "Rings": Process is defined by the "Rings". Quality Manager and Project Management Tool will also re-define some steps of the plan.

Goal 2: Information related to the use of the organization's standard software process by the software projects is collected, reviewed, and made available.

Solution in "Rings": The process will be tracked by the Software Project Management Tool and the results of the tool will be reviewed and checked by Quality Manager of management team.

Training Program

Goal 1: Training activities are planned.

Solution in "Rings": Training activities will be defined by the Surgeon and planned by the Project Manager, because most of the training programs include technical issues. The Training issues are not prepared detailed, but there is a breakpoint for training between the user stories and feedback phases.

Goal 2: Training for developing the skills and knowledge needed to perform software management and technical roles is provided.

Solution in "Rings": The detailed training programs may be defined by Surgeon and the process has necessary gaps for training breaks.

Goal 3: Individuals in the software engineering group and software-related groups receive the training necessary to perform their roles.

Solution in "Rings": The "Rings" methodology is a complete set. And for inner ring, there should be no training phase, because of the easiness of the process. And after each ring, the training phases should be considered.

Integrated Software Management

Goal 1: The project's defined software process is a tailored version of the organization's standard software process.

Solution in "Rings": Because the company starts at first ring of the "Rings" and grows according to the rings, the process of the development group is tailored version of the organization standard process.

Goal 2: The project is planned and managed according to the project's defined software process.

Solution in "Rings": The Project Manager and Project Management Tool are responsible by tracking the project about going according to the plans.

Software Product Engineering

Goal 1: The software engineering tasks are defined, integrated, and consistently performed to produce the software.

Solution in "Rings": Engineering tasks are defined in the "Rings" process list.

Goal 2: Software work products are kept consistent with each other.

Solution in "Rings": Engineering tasks are defined in the "Rings" process list.

Intergroup Coordination

Goal 1: The customer's requirements are agreed to by all affected groups.

Solution in "Rings": Project Manager and Documenter will attend to the meetings and they will get intergroup coordination.

Goal 2: The commitments between the engineering groups are agreed to by the affected groups.

Solution in "Rings": Documenter will document all of the phases and will let the groups checked by the customers.

Goal 3: The engineering groups identify, track, and resolve intergroup issues.

Solution in "Rings": Project Manager will control all process.

Peer Reviews

Goal 1: Peer review activities are planned.

Solution in "Rings": Review activities will be done by Pair Programming and Surgeon.

Goal 2: Defects in the software work products are identified and removed.

Solution in "Rings": A good pair programming application will solve most of the defects before testing phase.

3.3 Usability in Market

There are two defined important risks for "Rings". The first one is the approval of quality that "Rings" gathers to the company and the second one is the usability of "Rings" practically.

For the first risk, the Capability Maturity Model is used. The ideal application of “Rings” is tested via checklist of maturity model and the results shows that the methodology will improve the process quality of a development company.

For finding answers about the second risk, a survey and a questionnaire are applied to the possible users of the software market in Turkey.

3.3.1 Research Domain and General Comments

The research domain contains several experienced project managers, software engineers and software developers. The answers of them will be very meaningful for determining the usability and power of the new software development methodology in the real world. The subjects and their comments are listed in the alphabetical order:

Volkan ABUR: Volkan ABUR is Software Engineer-Analyst in IDEON. The company works on software development for defined companies. His specialty is on defining the requirements and the workflows of the customers to automate it with software and computer support. Because of his work in company, the discussions focus on requirements elicitation and analyze phases of “Rings”. Gathering user stories as elicitation is a useful idea for small projects, as result. Another good idea in “Rings” is also the incremental improvement of the product.

Hakan F. AKMERIÇ: Hakan AKMERIÇ is the general director of the Computing Center of KIRAÇA Holding. His background is on project management. Our first contact was done via e-mail. His first impressions were positive.

Taner AKSOY: Taner AKSOY is the general director and founder of GLOBAL Software. Their experience is especially about software on shipping and adaptation of software packets. The company has already 20 developers in software development and maintenance. Our first contact was done via e-mail. And in a meeting, it was seen that “Rings” has the process solutions that they have. His only

anxiety is about adaptation of the new software in the already running system on customer side.

Ilker ARABACI: Ilker ARABACI is the general director of DATATRAINING ETG. He is also the founder of the company. The company is specialized on education and developing special solutions. His first impressions are positive. The only deficiency according to Mr. ARABACI is handling network and hybrid projects. However, the first aim of “Rings” is handling software projects. So, this can’t be seen as a problem of “Rings”, but this may be a defect.

Köksal ATAMAN: Köksal ATAMAN is the general director of Project offices in BIMAR. He has experience of many years in project management. His company is founded to develop solutions on problems of ARKAS. In last 2 years, he is the head manager of the project “Changing process in software development of BIMAR”. The new software development process is Extreme Programming and their aim is taking CMM – 2.level. The first contact was obtained by email. His first impressions are positive. His only anxiety is about the practicality of several Extreme Programming ideas, like Pair Programming. At second contact, the usability of “Rings” was discussed in Ataman’s office. The discussion goes around the similarities between “Rings” and “Extreme Programming”. (Note: According to his opinions, these two methodologies are similar, but “Rings” has the solution of the “Coordinator” problem of Extreme Programming, which is important for controlled projects.)

Bilgem ÇAKIR: Bilgem ÇAKIR is developer (Software Engineer) in SOBEE. SOBEE is focused on entertainment software, 3D modeling and game development. Mr. ÇAKIR’s most important properties in this survey are his Research and Development experience and Rational Unified Process experience. According to his opinions, the most important value in “Rings” is the “time” parameter in the methodology. The build-in improvement of “Rings” is one of the most attractive ideas of thesis.

Mustafa DÖNMEZ: Mustafa DÖNMEZ is Project Manager in YAPI KREDI Bank. His experience is on information systems and project management. He has also programmer background in his previous companies. His first impressions are very positive. According to his e-mail, he is very impressed. However detailed impressions couldn't be taken. Because he is in Istanbul, discussions were made by e-mail. According to the opinions in the questionnaire, Mr. DÖNMEZ had found "Rings" useful and usable in the market.

Tamer GÜLCE: Tamer GÜLCE is Project Manager in TREDA. The main work areas of Treda are counseling software development companies and other companies, preparing intranet solutions, adaptation of software packages, etc. One of their last important points is accreditation of TREDA and counseling on accreditation. TREDA will be certified with ISO 9000 and CMM-3 in next days. His first impressions are very positive.

Onur GÜNDURU: Onur GÜNDURU is the Founder-Software Engineer of "ONUR Yazilim Ltd.". The company works on two project types: Special software for companies and game development. The discussions about "Rings" had gone about the similarities with Extreme Programming. Mr. GÜNDURU has the opinion that Agile Software Development Methodologies are more acceptable in Turkey's software market and the appropriation of the problems, like coordination, documentation or requirements elicitation is a good start. Additionally, the coverage of these problems is enough in "Rings".

Selim HENDRICKSON: Selim HENDRICKSON is software developer in IZTECH A.S. IZTECH A.S. works on standard software packets to control the money in companies. Because of the heavy work load of Mr. HENDRICKSON, the contacts were established via e-mail. Via these contacts, the positive impressions were taken.

Ufuk ILTER: Ufuk ILTER is the project manager of DIANA. This software company works especially on tourism software: Tourism portals, Hotel automation

software, etc. Mr. ILTER is the project manager of 10 software developers. The first contact is established in his office. His first impression is very positive. “Rings” is complete and highly usable in a software company. Additionally, they were used too similar processes in development cycles of two software packages.

Güner MUTLU: Güner MUTLU is the leader of software development group in BIMAR. BIMAR’s first job is preparing solutions for ARKAS. As a secondary aim, it can be seen that selling these solutions to other companies in the same market. Mr. MUTLU focused on the practicality of “Rings” for programmers in the discussion. The influence from Extreme Programming in analysis phase is useful and the documentation phase in first ring will be useful for new companies.

Koray OTAG: Koray OTAG is the general director and founder of DVP Interactive. The main works of DVP Interactive are selling FIDELIO products (adaptation in companies) and developing web solutions for companies. Koray OTAG is the director of Izmir office. In the first meeting, the opinions are that the process is completely usable for a new company. Additionally, DVP was used the “Middle Ring” in a project. The success of this project is also a proof that the system may be usable in a company. Details about this project had taken and the meetings were continued. At meetings and applied questionnaire, it can be seen, that DVP had partly used “Rings” methodology and the overall method is usable in a software development company, which is specialized in Web projects.

Murat ÖZEMRE: Murat ÖZEMRE is the Software Project Manager of BIMAR. BIMAR’s first job is preparing solutions for ARKAS. As a secondary aim, it can be seen that selling these solutions to other companies in the same market. In the discussions, impressions were good. The details of phases and the responsibilities of the team members are the center points of discussions. The defined “Rings” was seen enough and a good try for the starter projects of new companies.

3.3.2 Reactions of Market

The software development methodology, “Rings” is overall accepted by the software development market. The Project Managers and Software Engineers as the potential users of the system checked the methodology and their first impressions were positive. Additionally, a questionnaire was applied to these people. In questionnaire, some questions were asked to define the problems in their companies, which are originated by development methodology. As second, some questions were asked about the usability of “Rings”.

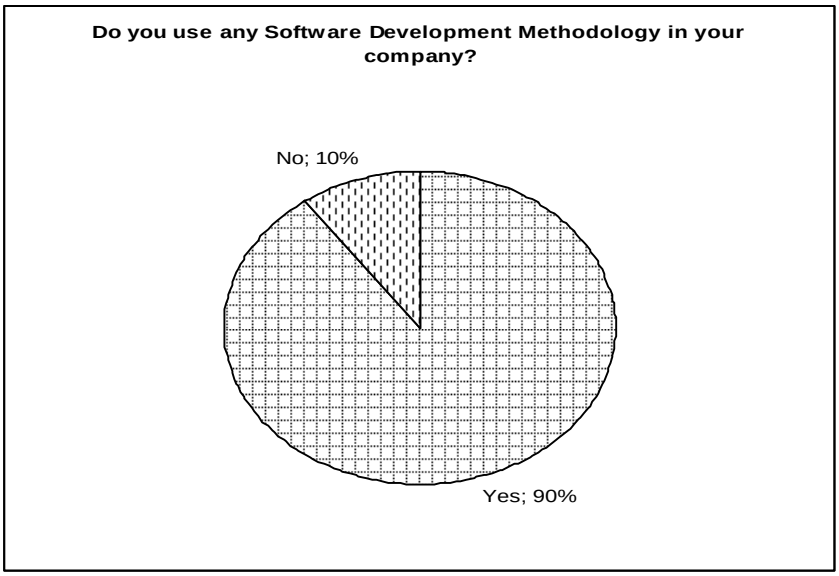

Figure 3.2 Are the subjects familiar with any Software Development Methodologies?

The software companies in Turkey work mostly on customer specific projects (not software for market) and most of these companies use their specific development processes. However, these companies have also many problems, which may be solved with right software development methodology. (Figure 3.2)

Table 3.1 If you should describe the Software Development Methodology in your company, which of the following basic phases can be defined?

Phase	Absolutely (%)	Often (%)	Rarely (%)	Absolutely Not (%)
Requirements A.	55	35	10	0
Specification	35	45	20	0
Design	55	23	11	11
Coding	100	0	0	0
Testing	55	45	0	0
Maintenance	45	33	22	0

Even the small companies in the survey, use most phases in the basic life cycle often. All of the companies have the agreement that they can specify and define their phases according to the definitions of Software Engineering. (Table 3.1)

Table 3.2 About Requirements Analysis

If one of your projects is thought,	Yes (%)	No (%)
Is there any team, which is only responsible to analyze User Requirements?	45	55
Are the User Requirements taken in the beginning of project?	90	10
Are the User Requirements taken in the beginning of each module?	70	30
Are there standard forms to gather User Requirements?	40	60
Do you have problems, while gathering User Requirements?	45	55
Are there any problems with gathered User Requirements?	75	25

When it is focused on Requirements Analysis phase, it can be seen the companies, who have the resource, separate their requirements analysis department. Another fact is that the companies want to gather the user requirements, while the project begins. But the requirements are not proved or the phase is not repeated in the module starts. The big companies mostly use standard forms for requirements definitions, but small companies, even if they prepare it in every project, don't use such documents.

It was seen that most of the companies have problems with requirements analysis. (Common reasons are the customer's limited knowledge and priority problems of customers) Big companies solve these problems, with additional coordinators, detailed documents and additional control systems. A problem in the survey is that the small companies work with requirements analysis phase and they have also problems with gathering needs of users. (Table 3.2)

Table 3.3 About Specification Phase

If one of your projects is thought,	**Yes (%)**	**No (%)**
Is there any team, which is only responsible to analyze User Requirements?	35	65
(if exist) Are the Specification Documents prepared by Computer Engineers?	60	40
Do you have standards for Specification?	65	35
Do you have problems, while preparing Specification Documents?	60	40
Are there any problems with prepared Specification Documents?	65	35

For Specification Phase, most of the companies use Computer Engineers and they don't have specialized teams for this job. Additionally, they use standard documentation. However, most of the companies often have problems about the specification. The problems are about the information transfer from users to developers. This transfer can't be done well with the defined specification phases. Another problem is also keeping the specification documents updated. (Table 3.3)

Table 3.4 About Design Phase

If one of your projects is thought,	**Yes (%)**	**No (%)**
Is there any team, which is only responsible to prepare Design?	25	75
Do you use any assistant tools?	65	35
Do you prepare entire design in the beginning of the project?	65	35
Do you prepare entire design in the beginning of each module?	85	15
Do you use design standards? (for example UML)	50	50
Do you have problems, while preparing Designs?	35	65
Are there any problems with prepared Designs?	65	35

In software development companies, there can't be seen a defined design team and the development groups prefer to use design tools if possible. The companies, which are focused on inner-projects don't want design standards, like Unified Modeling Language. However, the companies, who sell their products, try to use such standards. The companies say that they have not so many problems about design phase. But the managers bring that there will be problems, when they can't serve enough time for this phase. (Table 3.4)

Table 3.5 About Coding Phase

If one of your projects is thought,	**Yes (%)**	**No (%)**
Are the specialties of your Coding Team, Computer Engineers?	65	35
Do you have defined Coding Standards?	80	20
Do you have Software Heroes in your team?	45	55
Do you have general problems?	60	40

Table 3.6 About Testing Phase

If one of your projects is thought,	**Yes (%)**	**No (%)**
Is there any team, which is only responsible to do Testing?	35	65
Do you work with your customer, while preparing Test material?	90	10
Are the Test materials prepared in the beginning of the project?	40	60
Do you have standard forms for Testing?	45	55
Do you have problems, while preparing Tests?	90	10
Are there any problems with prepared Tests?	65	35

When focused on implementation and testing phases, it can be seen that the small companies use computer engineers for these phases and the guarantee of these phases are left to individuals. However, there are defined standards, documentations and reports for guaranteed result in bigger teams. Another fact from the survey is that the companies see that the customers are only responsible for preparation of the testing data. (Table 3.5, Table 3.6)

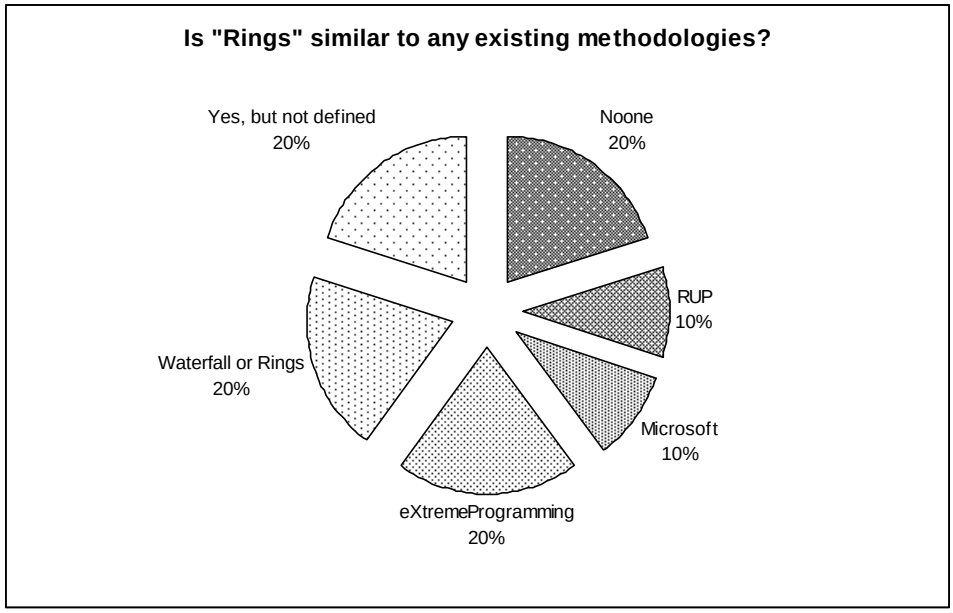

Figure 3.3 Similar Methodologies to “Rings”, according to the subject’s opinions

Original results have been obtained from the last section of the survey which is only about the proposed methodology, “Rings”. Some of the managers said that they didn’t see a methodology like “Rings”. Some said that the methodology is similar to Agile Methodologies and some to Heavyweight ones. This result has a certain meaning that the new methodology has something useful from the previously defined methodologies. Every subject see “Rings” how he/she wants to see: The developers who need a disciplined approach may use the system to improve his/her process more documented and controlled. However, the developers who want to use an approach for a controlled but not many documented system; they also see the system as an introduction to the agile software development methodologies. This situation shows that “Rings” is suitable for all development companies, either they need disciplined or not. (Figure 3.3)

Table 3.7 Have you ever thought that your company used any phases of “Rings”?

Phases (I. Ring of “Rings”)	**Used**	**Not Used**
User Stories	75	25
Version Planning	90	10
Interface – Database Design	90	10
Coding – Unit Tests	100	0
Integration – Feedback	100	0
Documentation	70	30
Phases (II. Ring of “Rings”)	**Used**	**Not Used**
Project Basics	75	25
Version Planning	80	20
Basic Interface – Database Design	90	10
Module based User Stories	25	75
Coding – Unit Tests	90	10
Integration – Feedback	90	10
Phases (III. Ring of “Rings”)	**Used**	**Not Used**
Project Basics	70	30
Resource Planning	85	15
Version Planning	85	15
Basic Interface – Database Design	85	15
Module based User Stories	35	65
Prototyping	25	75
Coding – Unit Tests	80	20
Integration – Feedback	80	20

When the phases of “Rings” were thought individually, the subjects agree most of the phases which were used in their companies; either they are mature or young companies. All of the subjects also agree on the usability on the methodology. (Table 3.7)

CHAPTER FOUR
CONCLUSIONS

4.1 Conclusion and Future Works

The software market is a market, from which the players can earn money easily and take place, because its product (software) is place-independent. The developers and customers may be in different places in the world and there can also be a trade between them, for the software is an invisible product, which may run on the wires of Internet.

However, the problems about the software also begin here. This invisible and place-independent product should be developed in a controlled way, because its cost is too much, which can't be wasted easily. The Software Development Methodologies take the place at this point. The existing methodologies are used to concretize this development process for better estimations, for higher product quality and for measuring it.

"Rings" is one of the studies for developing solutions for these problems in the software market. The most important difference of the new methodology than the existing ones is that the acceptance the rules of the nature of the software development and trying to apply the "applicable" software engineering practices to the projects. "Rings" is prepared with the real needs and applications of developers in the market and this property brings it one step nearby the developers than the existing ones.

This study explains the new methodology, "Rings", which may be named as an agile software development methodology, but it can be classified between Agile Software Methodologies and Heavyweight Software Development Methodologies. "Rings" starts as an Agile Methodology and improves the company to a Heavyweight Methodology according to the changing needs of the company.

“Rings” runs on a defined scenario of a young software company. 1st ring of the methodology is suitable for a new company, which has not so much budget and time for completing the project. In 2nd ring, the suggestions about the company are more resources and more responsibility for the successfully completed project. The 3rd ring is prepared for more critical projects. (Therefore the company has more responsibility about the projects.)

This study contains entire information which may be needed for the application in the company. The contents of the documents in the rings, the responsibilities and the required skills of team members and the completion criteria of the phases were explained with details in the sections for easier implementation on the company.

After this study, the “Rings” may be inspected on real practices. The survey on companies has the results that “Rings” is not too different than the phases that the subjects already use. So, as an additional work, “Rings” will be announced to the possible users and the results of application of the methodology may be the first and most important future work. As secondary, a software engineering tool may be developed for assistance to the project managers, who want to apply the “Rings”. This tool may help the companies for helping on the application of the methodology and for helping on the decision of the ring transitions.

As result, the development methodologies will help to the Turkish companies in the world software market to develop world-wide products. “Rings” will be a good and different study for the companies, who want to go from a local software developer to a world-wide software development company.

REFERENCES

Agile Alliance, *Principles of agile manifesto* (n. d.). Retrieved April 12, 2006, from http://www.agilealliance.org.

Beck, K. (1999). *Extreme programming explained*, Addison-Wesley Pub. Co., 0201616416.

Birant, K.U. & Kut, R. A. (2005). *Pair programming in XP*, Biltek 2005, International Informatics Congress, 11.06.2006.

Boehm, B. (1988). *A spriral model of software development and enhancement*, IEEE 1988, p 61 – 72.

Chrysler comprehensive compensation system (n. d.). Retrieved April 12, 2006, from http://www.wikipedia.com.

Cockburn, A. & Williams, L. (2000). *The costs and benefits of pair programming*, Extreme Programming and flexible Processes in Software Engineering, XP2000.

Endres, A. & Rombach, D. (2003). *A handbook of software and systems engineering*, Addison-Wesley Pub. Co., 0321154207.

Ferzli, M., Wiebe, E. & Williams, L. (2002). *Paired programming project: Focus groups with teaching assistants and students*, NCSU Technical Report, TR-2002-16 November 25, 2002.

Fowler, M. (13 December 2005). *The new methodology*, Retrieved April 12, 2006, from http://www.martinfowler.com.

Ghezzi, C., Jazayeri, M. & Mandrioli, D. (1991). *Fundamentals of software engineering*, Prentice Hall Inc., 0138182043.

Highsmith, J. (2002). *Agile software development ecosystems*, Addison Wesley Pub. Co., 0201760436.

Hohmann, L. (2003). Beyond software architecture, Addison-Wesley Pub. Co., 0201775948.

Jeffries, R. (2001). *What is extreme programming?*, Retrieved April 12, 2006, from http://www.xprogramming.com.

Keefer, G. (2003). *Extreme programming considered harmful for reliable software development 2.0*, AVOCA GmbH.

Paulk, M., Curtis, B., Chrissis, B. & Weber, C. (1993). *Capability maturity model for software*, Technical Report, Carnegie Mellon University Software Engineering Institute, CMU/SEI – 93-TR-24.

Pressman, R. (2001). *Software engineering* , Mc-Graw Hill Pub. Co., 0073655783.

Royce, W. (1970). Managing the development of large software systems, *IEEE Wescon*, August 1970, p 328-337.

Saridogan, M.E. (2004). *Yazilim mühendisligi*, Papatya Yayinlari, 9756797576.

Schach, S. (2002). *Object-oriented and classical software engineering*, McGraw-Hill Pub. Co., 0072395591.

Sommerville, I. (1992). *Software engineering*, Addison-Wesley Pub. Co., 0201565293.

Sommerville, I. & Sawyer, P. (1997). *Requirements engineering: A good practice guide*, Wiley&Sons Pub., 0471974447.

Von Mayrhauser, A. (1990). *Software engineering methods and management*, Academic Press Inc., 0127273204.

Wells, D. (17 February 2006). *XP practices and rules*, Retrieved April 12, 2006, from http://www.extremeprogramming.org.

Williams, L. & Upchurch, R. (2001). Extreme programming for software engineering education, *31st ASEE/IEEE Frontiers in Education Conference* 2001.

Williams, L., Yang, K., Wiebe, E., Ferzli, M. & Miller, C. (November 2002). Pair programming in an introductory computer science course: Initial results and recommendations, *OOPSLA Educators Symposium 2002*.

APPENDICES

A. Chrysler Comprehensive Compensation System (C3)

The Chrysler Comprehensive Compensation System (commonly referred to as 'C3') was a project in the Chrysler Corporation to replace several payroll applications with a single system. The new system was built using Smalltalk and Gemstone. The software development techniques invented and employed on this project are of interest in the history of software engineering. C3 has been referenced several books on the Extreme Programming methodology. (Wikipedia, n. d.)

The C3 project started in 1995. The end goal was to build a new system to support all the payroll processing for 87,000 employees by 1999 (Keefer, 2003). About a year later Kent Beck was hired to get the thing working, at this point the system had not printed a single paycheck.(Keefer, 2003) Beck in turn brought in Ron Jeffries. In March 1996 the development team estimated that the system would be ready to go into production around one year later. In 1997 the development team adopted a way of working which is now formalized as Extreme Programming. (Highsmith, 2002) The one-year delivery target was nearly achieved, with the actual delivery being a couple of months late; the small delay being primarily due to lack of clarity regarding some business requirements. A few months after this first launch, the project's customer representative — a key role in the Extreme Programming methodology — quit due to severe burnout and stress, and couldn't be replaced. (Keefer, 2003)

The plan was to roll out the system to different payroll 'populations' in stages, but C3 never managed to make another release despite two more year's development. The C3 system only ever paid 10,000 people. (Hendrickson, 2001) Performance was something of a problem; during development it looked like it would take 1000 hours to run the payroll, but profiling activities reduced this to around 40 hours; another month's effort reduced this to 18 hours and by the time the system was launched the

figure was 12 hours. During the first year of production the performance was improved to 9 hours.

Chrysler was bought out by Daimler-Benz in 1998, after the merger the company was known as DaimlerChrysler. DaimlerChrysler stopped the C3 project on 1 February 2000. (Keefer, 2003)

Frank Gerhart, a manager at the company, announced to the XP conference in 2000 that DaimlerChrysler had de facto banned XP after shutting down C3. However some time later DaimlerChrysler resumed the use of XP.

B. The Rings Questionnaire

A questionnaire is applied to project managers and software developers to software market for taking their opinions about problems of existing software development systems and usability of “RINGS”. The questionnaire may be found as follows; (The language of questionnaire is Turkish.)

DOKUZ EYLÜL ÜNIVERSITESI BILGISAYAR MÜHENDISLIGI BÖLÜMÜ

Anket Çalismasi

Yazilim Gelistirme Süreçleri
("Çemberler" ("Rings"))

HAZIRLAYAN:
Aras. Gör. Kökten Ulas BIRANT (Bilgisayar Y. Müh.)
Dokuz Eylül Üniversitesi Bilgisayar Mühendisligi Bölümü
Tinaztepe Yerleskesi Buca/Izmir
ulas@cs.deu.edu.tr
http://cs.deu.edu.tr/ulas

Bu anket çalismasi, Dokuz Eylül Üniversitesi Bilgisayar Mühendisligi Bölümünde sürdürülmekte olan "Yeni Bir Yazilim Gelistirme Süreci: Rings" konulu Doktora çalismasina karsi piyasa tepkilerinin tanimlanmasi amaciyla hazirlanmistir. Anketin hedef kitlesi yazilim üretiminde çalismis ve/veya çalismakta olan Proje Müdürlügü, Yazilim Mühendisligi, Yazilim Gelistiriciligi, Kodlayicilik, vb. ünvanlardaki kisilerdir.

Çalismanin ilk bölümünde anketi dolduran kisinin kendisi ve çalistigi sirket konusunda temel bilgiler alabilmek amaciyla tanitici bilgiler sorulmaktadir.

Çalismanin ikinci bölümünde anketi dolduran kisinin sirketinde kullanilan yazilim gelistirme sürecinin tanimlanmasi amaciyla sorular sorulmaktadir.

Çalismanin son bölümünde bilgileri paylasilmis olan "Çemberler" ("Rings") Yazilim Gelistirme Süreci hakkinda anketi dolduran kisinin fikirleri sorulmaktadir.

Elinizdeki anketin son bölümü, terimlerle ilgili karmasanin önlenmesi amaciyla Yazilim Mühendisligi terimlerinin tanimlanmasina ayrilmistir.

Anketin dogru sekilde ve hedef kitlesi tarafindan doldurulmasi, tez sonucunda olusturulacak yazilim sürecinin yazilim piyasasinda uygulanabilir olarak yayinlanmasi konusunda çok önemlidir. Anketimizi doldurarak yaptiginiz yardim için tesekkür eder, bölümümüzde yapilacak etkinliklerde de sizleri aramizda görmekten mutlu olacagimizi belirtirim.

Saygilarimla,

Prof.Dr. R. Alp KUT
Dokuz Eylül Üni. Bilgisayar Müh. Blm. Bsk.
Tez Danismani

Iletisim ve her türlü soru için:
Kökten Ulas BIRANT (Doktora tez ögrencisi)
Dokuz Eylül Üniversitesi Bilgisayar Mühendisligi Bölümü
Tinaztepe Yerleskesi Buca/IZMIR
ulas@cs.deu.edu.tr
http://cs.deu.edu.tr/ulas

Tanitici bilgiler

Firma Adi:

Web adresi:

Kurulus yili: (bilgi islem/yazilim alaninda)

Çalisma alani:

Adiniz:	
Son mezun oldugunuz okul:	
Mesleginiz:	
(varsa) Departmaniniz:	
Göreviniz/Isiniz:	
Is tecrübeniz:	
(son 3)	

Dokuz Eylül Üniversitesi Bilgisayar Mühendisligi Bölümünde hazirlanacak projelerden ve/veya etkinliklerden haberdar olmak ister misiniz?

a. Evet, bilimsel ve/veya proje bazli etkinliklerden haberdar olmak isterim.

b. Hayir, istemiyorum.

Süreç Tanimlayici bilgiler

Yazilim projelerinizi nasil tanimlarsiniz? a. Müsteriye özel projeler (Tanimli kullanicilar) b. Piyasaya yönelik projeler (Tanimlanmamis kullanicilar)

Sirketinizde bir yazilim gelistirme süreci uygulaniyor mu?		a. Evet	b. Hayir
Cevabiniz "Evet"se	a. Bilinen bir süreç	b. Sirketimize özel	

Sirketinizdeki Yazilim Gelistirme Sürecini tanimlamaniz gerekirse asagidaki temel yazilim gelistirme süreci adimlarindan hangilerini uyguluyorsunuz?				
Adim	**Mutlaka**	**Siklikla**	**Nadiren**	**Hayir**
Gereksinim Analizi				
Belirtimleme				
Tasarim				
Kodlama				
Sinama (Test)				
Bakim				

Bir yazilim projenizi düsünecek olursak ...	Evet	Hayir
Görevi sadece müsteri gereksinimlerini toplamak olan bir ekibiniz var mi?		
Müsteri gereksinimleri proje baslangicinda mi aliniyor?		
Müsteri gereksinimleri modül baslangicinda mi aliniyor?		
Müsteri gereksinimleri toplamak için standart formlariniz mevcut mu?		
Müsteri gereksinimlerinin toplanmasinda sorunlar yasiyor musunuz?		
Toplanan müsteri gereksinimleri ile ilgili sorunlar (eksik/hatali/...) yasiyor musunuz?		
Müsteri gereksinimleri ile ilgili (varsa) sorunlarinizi özetleyebilir misiniz?		

Bir yazilim projenizi düsünecek olursak ...	Evet	Hayir
Görevi sadece Belirtimleme Raporlarini hazirlamak olan bir ekibiniz var mi?		
(varsa) Belirtimleme Raporlarinizi Bilgisayar Mühendisleri mi hazirliyor?		
Belirtimleme Raporu Standardiniz var mi?		
Belirtimleme Raporu hazirlanmasinda sorunlar yasiyor musunuz?		
Hazirlanan Belirtimleme Raporu ile ilgili sorunlar (eksik/hatali/...) yasiyor musunuz?		
Belirtimleme Raporlari ile ilgili (varsa) sorunlarinizi özetleyebilir misiniz?		

Bir yazilim projenizi düsünecek olursak ...	Evet	Hayir
Görevi sadece tasarim hazirlamak olan bir ekibiniz var mi?		
Tasarim hazirlarken yazilim araci kullaniyor musunuz?		
Tasarimlar proje baslangicinda mi hazirlaniyor?		
Tasarimlar modül baslangicinda mi hazirlaniyor?		
Tasarim standartlari kullaniyor musunuz? (örnegin UML)		
Yazilimin tasarlanmasinda sorunlar yasiyor musunuz?		
Hazirlanan tasarimlar ile ilgili sorunlar (eksik/hatali/...) yasiyor musunuz?		
Yazilim Tasarimi ile ilgili (varsa) sorunlarinizi özetleyebilir misiniz?		

Bir yazilim projenizi düsünecek olursak ...	Evet	Hayir
Kodlayici ekibiniz Bilgisayar Mühendislerinden mi olusuyor?		
Tanimli Kodlama Standartlariniz var mi?		
Ekibinizde "Yazilim Kahramanlari*" var mi?		
Müsteri gereksinimleri toplamak için standart formlariniz mevcut mu?		
Kodlama ile ilgili sorunlar (uzun/hatali/...) yasiyor musunuz?		
Kodlama çalismaniz ile ilgili (varsa) sorunlarinizi özetleyebilir misiniz?		

Bir yazilim projenizi düsünecek olursak ...	Evet	Hayir
Görevi sadece sinama olan bir ekibiniz var mi?		
Sinama verileri hazirlanirken müsteri ile çalisma yapiliyor mu?		
Sinama verileri proje baslangicinda mi hazirlaniyor?		
Sinama verilerini tanimlamak için standart formlariniz mevcut mu?		
Sinama verilerinin tanimlanmasinda sorunlar yasiyor musunuz?		
Tanimlanan sinama verileri ile ilgili sorunlar (eksik/hatali/...) yasiyor musunuz?		
Sinama (Test) ile ilgili (varsa) sorunlarinizi özetleyebilir misiniz?		

Bir yazilim projenizi düsünecek olursak ...	Evet	Hayir
Görevi sadece yazilim bakimi olan bir ekibiniz var mi?		
Bakim yapan gelistiriciler, ayni zamanda baska projelerde de çalisiyorlar mi?		
Bakim isteklerinin, gereksinimlerle çelistigi durumlar siklikla yasaniyor mu?		
Müsteri bakim isteklerini almak için standart formlariniz mevcut mu?		
Müsteri bakim isteklerinin toplanmasinda sorunlar yasiyor musunuz?		
Alinan bakim istekleri ile ilgili sorunlar (eksik/hatali/...) yasiyor musunuz?		
Müsteri gereksinimleri ile ilgili (varsa) sorunlarinizi özetleyebilir misiniz?		

"Çemberler" ("Rings") hakkinda

"Çemberler" , uyguladiginiz veya incelediginiz süreçlere benziyor mu? (Evet'se, hangilerine?) a. Hayir b. Evet, ...

Çemberler'in bir veya daha fazla fazini **genel olarak** uyguladiginizi düsündünüz mü?	Iç çember	Orta çember	Dis çember

Iç çember'i uyguladiginizi düsündüyseniz, hangi adimlari uyguladiniz?		
Adimlar	**Uygulandi**	**Uygulanmadi**
Kullanici Hikayeleri		
Sürüm Planlama		
AY-VT Tasarimi		
Kodlama – Birim Sinamalari		
Entegrasyon – Geri bildirimler		
Raporlama		

Orta çember'i uyguladiginizi düsündüyseniz, hangi adimlari uyguladiniz?		
Adimlar	**Uygulandi**	**Uygulanmadi**
Proje Temelleri		
Sürüm Planlama		
Temel AY-VT Tasarimi		
Modül Temelli Kullanici Hikayeleri		
Kodlama – Birim Sinamalari		
Entegrasyon – Geri bildirimler		

Dis çember'i uyguladiginizi düsündüyseniz, hangi adimlari uyguladiniz?		
Adimlar	**Uygulandi**	**Uygulanmadi**
Proje Temelleri		
Kaynak Planlama		
Sürüm Planlama		

Temel AY-VT Tasarimi		
Modül Temelli Kullanici Hikayeleri		
Prototipleme		
Kodlama – Birim Sinamalari		
Entegrasyon – Geri bildirimler		

Çemberlerin uygulanabilirligi konusunda ne düsünürsünüz?

Iç çember		
Adimlar	**Uygulanabilir**	**Uygulanamaz**
Kullanici Hikayeleri		
Sürüm Planlama		
AY-VT Tasarimi		
Kodlama – Birim Sinamalari		
Entegrasyon – Geri bildirimler		
Raporlama		
Orta çember		
Adimlar	**Uygulanabilir**	**Uygulanamaz**
Proje Temelleri		
Sürüm Planlama		
Temel AY-VT Tasarimi		
Modül Temelli Kullanici Hikayeleri		
Kodlama – Birim Sinamalari		
Entegrasyon – Geri bildirimler		
Dis çember		
Adimlar	**Uygulanabilir**	**Uygulanamaz**
Proje Temelleri		
Kaynak Planlama		
Sürüm Planlama		
Temel AY-VT Tasarimi		
Modül Temelli Kullanici Hikayeleri		
Prototipleme		
Kodlama – Birim Sinamalari		
Entegrasyon – Geri bildirimler		

Tanimlamalar

YAZILIM KAHRAMANI: Yazilim sirketinde; süreçlere hakim, kodlama konusunda tecrübeli, genel Bilgisayar Mühendisligindeki son teknolojik gelismelere hakim, problem çözümleme konusunda yetenekleri diger takim üyelerince de kabul edilen, projede çikabilecek herhangi bir problemde çözmesi beklenen proje personelidir.

STANDART YAZILIM GELISTIRME SÜRECI: 60'li yillarda ilk kez Yazilim Mühendisligi kavraminin ortaya atilmasi ile birlikte varolan Mühendislik uygulamalarindan esinlenilerek hazirlanan ve yazilim hazirlanmasinda temel olarak kabul edilen gelistirme adimlaridir.

a. GEREKSINIM ANALIZI: Yazilim projesinde müsteri isteklerinin toplandigi, incelendigi, raporlandigi ve bu isteklerle ilgili sorunlarin çözümlendigi adimdir.
b. BELIRTIMLEME: Müsteri isteklerinin raporlanmasinin ardindan müsteriyle görüsmeler sonucunda müsteri ve gelistirici tarafindan onayli mutabakat raporu hazirlanmasi adimidir.
c. TASARIM: Olusan müsteri isteklerine uygun olarak çözümün kodlama öncesinde çesitli çözüm anlatim dilleriyle ifade edildigi ve üzerinde tartisilarak çözümün nihai haline ulasildigi adimdir.
d. KODLAMA: Olusturulan tasarimin bilgisayar kodlarina dönüstürüldügü adimdir.
e. SINAMA (TEST): Olusturulan kodlarin ve tüm sistemin, çalisma ve kabul edilebilirlik açilarindan kontrolünün yapildigi adimdir.
f. BAKIM: Yazilimin müsteriye tesliminden sonra çikabilecek hatalarin düzeltildigi, gelisen teknoloji ile birlikte gerekli olan gelismelerin yapildigi adimdir.