CHAPTER 1

Introduction

1.1 Software Engineering

Software Engineering is a systematic, disciplined and structured approach which is used to develop and maintain software products. This signifies that if a different category of people employs same technique during the development of the software, the similar software will be obtained. It focuses on software to develop efficient software products that meet the constraints of cost of the software, quality of the software and time consumed by the software. System Development Life Cycle with respect to Software engineering relates to the process of composing or formulating systems, models, and methodologies which are used in development. Software Development Life Cycle provides a sequence of activities for software developers in order to develop the software such that it is completely built within the allotted time and the quality of the software is followed as per the standards imposed by the developers. Requirement prioritization in context with software engineering is elucidated in the Fig 1.1.

1.2 SDLC

It is a process used to design, develop and to produce high quality software. It produces quality software that meets customer requirements with on time delivery. The various
phases involved in software development are Requirements gathering and Analysis, Designing, Coding, Testing, and Implementation. A development team chooses the suitable Software Development Life Cycle (SDLC) model to be imposed with respect to the user requirements or application. The most challenging part is selecting most suitable model based on nature of the project and various technical, organizational, project and team considerations (Mishra and Dubey 2013).

1.3 Phases of SDLC

The process of software development is divided into structured and disciplined sections called as phases. The phases of SDLC are elucidated in the Fig 1.2. The most common phases in software development life cycle model are:

1. Requirement Gathering and Analysis
2. Designing
3. Coding
4. Testing
5. Maintenance and Support

Fig. 1.2 Phases of SDLC
The first step in SDLC is requirement gathering and analysis phase which helps the developer to understand the customer’s needs in terms of the problem definition. The purpose of the design phase is to plan a solution of the problem specified by the requirement document. The implementation of planned solution for the problem definition is done in coding phase, in testing phase the implemented solution to the problem domain is tested against various test cases and in maintenance and support phase the developed system is provided with support for optimal execution. There are numerous software life cycle models that are used for software development approach which are designed to develop software products.

1.4 Classical models of software development life cycle

1.4.1 Waterfall model

Waterfall model was stated by Royce in 1970 and is also called as the classical and linear-sequential life cycle model of Software Engineering (Suresh Kute and Thorat 2014). Every phase is executed in a sequential or linear manner i.e. one after the other due to which waterfall model cannot be assigned to dynamic projects. Various phases

![Fig. 1.3 Waterfall model](image-url)
Advantages of Waterfall Model are:

1. Waterfall model is simple and easy to understand.
2. Error generation is prevented by verification and validation.
3. Milestones are defined in prior for each phase.

Disadvantages of Waterfall Model are:

1. Requirements are frozen, i.e. once the set of requirements are determined then we cannot change it through the process.
2. Reverse tracking between phases is not possible.
3. Large, complex and tedious life cycle model.

Areas of Usage of Waterfall Model are:

1. Widely used in areas where requirements well understood and are frozen.
2. Short Duration software projects.
3. Automation of previously existing manual system is done.

1.4.2 Iterative Model

This model does not start with the full requirement specification. Instead, it begins by implementing just part of the specification, which is then reviewed later for further development. This process then repeats its iteration, producing a new version of the software for each cycle of the model. Each iteration follows waterfall model wherein feedback from one phase is provided to another phase. The software products that are obtained at series of steps are sent for production as incremental releases. The various phases that are involved in iterative model are elucidated in the Fig 1.4.

Advantages of Iterative Model are:

1. Projects developed under Iterative model can measure their progress.
2. Less cost is involved in modification of requirements
3. During execution of each iteration different results are obtained.
Disadvantages of Iterative Model are:

1. During the process of defining the increments the complete systems knowledge is required.

2. Unsuitable for smaller projects.

3. Large, complex and tedious life cycle model.

Areas of usage of Iterative Model are:

1. Development of incomplete knowledge.

2. Large projects with feedback providing mechanism at each phase.

3. Where new version of software is required for each release.

1.4.3 Spiral Model

Barry Boehm stated Spiral model in 1998 and its focuses are on risk analysis but it resembles the incremental model of development. There are four phases that are carried out by spiral model i.e. planning, risk analysis, engineering, and evaluation as elucidated in the Fig1.5. In order to successfully develop a project in the spiral model, the project should pass all the four phases repeatedly in iterations. At the initial stage, a complete plan is built. After planning, risk assessment is done for requirements that have been gathered. After which identification of risks and its respective solutions are suggested during risk analysis phase and as a result prototype is produced. At last the software
Advantages of Spiral Model:

1. Risk analysis is performed at a larger amount.
2. Earlier development of software.
3. Failure rates associated with the projects are less.
4. Termination of the product development can take place after any spiral being developed and the end product produced at that spiral corresponds to a working software.

Disadvantages of Spiral Model:

1. Estimation of cost and time is a tedious process.
2. Risk analysis cost higher, therefore, it is not suitable for small scale projects.
3. Highly expertized engineers are required to perform risk analysis.
4. As spiral models main concern is risk analysis the success factor depends on the analysed risk.
Area of Usage of Spiral Model:

1. Projects, where there is involvement of medium to high risks, takes place.
2. Projects where changes are welcomed during the development process.

1.4.4 V-Shaped model

![Fig. 1.6 V-Shaped model](image)

V shaped model is a sequential model in which a phase must be concluded, to begin with, the execution of next phase (Dholakia and Mankad 2013). The process of testing is done along with development phase which implies that testing of earlier phases is verified later as elucidated in Fig 1.6.

Advantages of V-Shaped Model:

1. In this model planning and understanding, the complexity is easy.
2. Defects found during testing can be back tracked and rectified at early stages.
3. The testing phase is executed before coding phase thus a lot of time is saved.
4. This model provides a higher chance of success while comparing it with waterfall model.

Disadvantage of V-Shaped Model:

1. Flexibility is low as compared with other models of development.
2. If any changes are welcomed during any of the stages then the requirements document needs to be updated.

3. Risks associated with the projects are higher.

Area of usage of V-shaped Model:

1. For Projects having small and medium size and those having fixed requirements which are clear.

2. Where expertise is available and technical resources are ample.

1.4.5 Agile software development model

![Agile software development model](image)

**Fig. 1.7** Agile software development model

The agile development team introduced agile software development method model in 2001 through the agile manifesto. It is a mixture of iterative and incremental process models which focus on customer satisfaction by early and continuous delivery of working software product as elucidated in the Fig 1.7. The main characteristics of this model are:

1. Incremental: There are Small software releases which are accompanied by rapid development cycles

2. Co-operative: Excellent customer-developer interaction.

3. Adaptive: An agile model is flexible enough to accommodate changes at the last moment.
Advantages of Agile Model:

1. Agile model has quick releases.
2. This model is sustainable to changing requirements
3. Gives more care on customer feedback
4. Measures real-time progress
5. Bad designs and faulty requirements are discovered and discarded immediately.

Disadvantages of Agile Model:

1. It is difficult for this model to use complex projects
2. Effective when the team is short.

Area of usage of Agile Model:

1. Used by self-organizing teams for effective building of their project.
2. Used in areas where there is a frequent change in requirements.

1.5 Requirement Gathering Analysis

The Requirement analysis is the first and the most important phase of the software development life-cycle SDLC. This phase translates the unclear, incomplete requirements of the users of software into complete and clear specifications. This clear specification of user requirements are useful in two ways, first it acts as a contract between the user and developer and secondly it is used in reducing software errors that occur during early stages of the development of the software thereby creating an effective software product.

1.6 What is a Requirement

What does a requirement mean in terms of software engineering? IEEE 610.12- 1990 standard has given a formal definition of a requirement. It defines a requirement as:

(a) To achieve the desired objective the user may need a condition or capability these conditions or capabilities are called as requirements.

(b) The conditions which must be possessed by the system which has to satisfy a standard or a specification.

A requirement is something which a user need or certain conditions that a system is required to meet or possess.
1.7 The Cost of Fixing Errors

A software development life cycle has some determined phases: requirement and analysis, design, coding, development testing, acceptance testing, and operation as elucidated in the Fig 1.8. If the errors are not fixed in the same phase where it has been caught then it will cost more to fix that error in later phases, for example, if a programming error which has not been detected until the acceptance testing phase then the cost of fixing the error becomes ten times the original cost of fixing the errors. Same way if a requirement error has not been detected until after the software has been delivered, and then the cost of fixing the error becomes a hundred times the original cost. Therefore, if an error is discovered soon then the cost of fixing the error is much cheaper.

1.8 Requirements elicitation and gathering

Elicitating requirements are the first step in the Requirements analysis process. The purpose of requirement elicitation is to find out what all problems need to be clarified (Finkelstein 1993). The factors on which elicitation techniques depends are:

1. Time factor.
2. Resources available.
3. The type of information that is needed to be elicited.

The other factor arises with the aspects as elucidated in the Fig 1.9.
1.9 Elicitation techniques

1.9.1 Brainstorming

Brainstorming produces numerous creative ideas which is constrained to a time limit, making the process visual, designating a facilitator, organizing a group with size (optimal group capacity may be 6-8), building ground rules and using the criteria to evaluate ideas.

The ground rules imposed for brainstorming are:

1. Rating the ideas and evaluating the criteria.
2. Criticism is not entertained.
3. There should be limited discussion and evaluation.

1.9.2 Document Analysis

Document analysis uses already existing documentation in order to elicit information. Document analysis serves the best when subject matter experts are not present in the organization. It is useful in understanding documentation of relevant details such as business rules, plan, contract, emails, and project charters.

1.9.3 Focus Groups

It is a well-structured and skilled process. A moderator is required from a select group who engages discussion, raises open questions and coordinates all members in eliciting information. As there is less number of individual interviews conducted it is cost and time saving process.
1.9.4 Interviewing

Interviewing is the process of asking questions in order to gear up ideas towards uncovering information.

The type of questions may be:

1. Formal or informal question.
2. Pointing towards an individual or selected group.
3. Maintaining focus towards the goals of the interview.
4. Open-ended questions which are used to find information and gaps.
5. Closed-ended questions to confirm and validate.

1.9.5 Prototyping

Prototyping focuses on the complete solution and it is the best approach for validation of requirements and used to uncover gaps. Time is consumed in searching for a solution to “how’s” rather than that of “what’s.”

Prototyping Visually represents the user interface that has:

1. Measures with good validation.
2. Great interaction amongst them.
3. Support for the visual learners.

1.9.6 Requirements Workshops

It is a systematic and facilitated event for all the stakeholders under a common platform to exchange their views, discover, prioritize then validate those requirements. The duration of a workshop can be a day or more. An experienced neutral facilitator is used to manage sessions during workshops.

Constraints in requirement workshop are:

1. Requirements should be elicited in a short span of period.
2. When many users participate they slow down the process.
3. If participants are fewer gaps are caused.
4. Availability of team members causes an issue.
1.9.7 Survey/Questionnaire

Questionnaires are used to elicit information during early stages of development. Effective Questionnaires include aspects and well-established domain boundaries. Both the developer and the participants must have complete knowledge about the Questionnaire. It should mainly be focused on avoiding redundancy and unnecessary information.

1.10 Types of Requirements

The following are the different types of requirements that are related to technical management. The main categorization of the requirements are as elucidated in the Fig 1.10.

![Fig. 1.10 Types Of Requirements](image)

1.10.1 Customer Requirements

Customer requirements mean the specification or desirable characteristics present in the product. There are two types of customer requirements

1.10.1.1 Service requirements

These are the intangible aspects that need to be fulfilled for a customer during purchasing a product. These aspects include timed delivery of products, making the payment easy, servicing the customer. This comprehends how a customer should be treated while buying a product or ordering a product and also about how his buying process goes. For example, if we are ordering a burger from McDonalds, then its on time delivery of food becomes service requirements.
1.10.1.2 Output Requirements

These are the tangible characteristics, features, specifications needed to be fulfilled to the customer while delivering the product. If the customer is seeking a service as a product it becomes output requirements. For example, consider an earphone, its product specification like the clarity of music and phone calls become its output requirements.

1.10.2 Functional Requirements

Functional requirements explain the set of inputs, their behaviours, and the outputs. Functional requirements involve data manipulation, calculations, and processing other functionality that define what a system should do. For example, Authentication, certification requirements, authorization levels, audit tracking etc.

1.10.3 Non-functional Requirements

The non-functional requirements are the constraint which is being imposed on the system due to internal and external factors. For example, Performance, Scalability, Capacity, Maintainability etc.

1.10.4 Performance Requirements

Performance requirements describe how well and what the system should perform. For example, The CPU utilizes 50% of the performance and then leaves other 50% for background jobs.

1.10.5 Design Requirements

The requirements that a design must accomplish in order to provide a efficient software. The design requirements design is of two categories:

1. By inclusion (build it this way), or
2. By exclusion (don’t build it this way).

For example, in designing a bat, the design requirements associated with the bat are:

1. The bat should weigh less than 1.5 pounds.
2. Should be manufactured by the given quality.

1.10.6 Derived Requirements

Derived requirements are indirect requirements or altered from higher-level requirement. For example, a system which requires wider range requirement or high speed
requirement may result in a change in the design requirement for low weight.

1.10.7 Allocated Requirements

A higher level requirement is partitioned and distributed to many lower level requirements. Example: A 50-kg item that has two sub-modules which might result in weight requirement of 20 kg and 30 kg for the two lower-module items.

1.11 Requirements prioritization

The below context defines requirement prioritization with respect to different authors:

1. Requirements prioritization as the process during which there is the identification of the most crucial requirements is done.

2. (Firesmith 2004) illustrates requirements prioritization as the process in which the orders of implementation of requirements are determined.

3. Frederick P. Brooks illustrates requirement prioritization as The difficult part of constructing a software system which is done by deciding clearly what needs to build.

All authors have defined requirements from their point of view and each differs from the other. Now, the question arise which one is correct and which one to follow.

1.12 What is Requirements Prioritization

The software system is composed of hundreds or thousands of software requirements. Due to limited resources in terms of time, budget, quality and other resources which are considered necessitous during the development of software and to get customer satisfaction we need to prioritize software requirements which are done by Requirement Prioritization as elucidated in the Fig 1.11. As all the elicited requirements cant be programmed in a single release and developers are not aware of which requirement is of higher priority in terms of the customer. In software development, if there is the participation of only one stakeholder it is easy to recognize high priority and less priority requirement but if stakeholders increase in number then it becomes complex to evaluate high and low priority requirements. The requirement which is of greater importance to one stakeholder might not be the same for other stakeholders therefore different stakeholders have different views on the requirements. Requirement prioritization creates a medium for different stakeholders to freeze on the requirements that are needed to be implemented during their respective releases. For example, In order to achieve the functionality of the high-value requirement A, requirement B, needs to be implemented first.
even though requirement B is of low value. System developers use requirement prioritization in order to implement those requirements which are to be implemented in the first release. The method of handling requirements differs from domain to domain and from company to company, therefore there is no correct way of prioritizing requirement.

1.13 Usefulness of requirements prioritization

Requirement prioritization not only avoids least important requirements but also is used to develop most adequate project management. To produce conflict solution amongst requirements and making decision making during architectural design and test case prioritization there is need of well-stated requirements (Herrmann and Daneva 2008). Thus, Harwell (Harwell et al. 1993) described a priority as an attribute of a requirement that is used for different aspects, depending on program and company needs. If all the requirements are categorized to be equally important, then it is hard for the project manager to respond towards budget cuts, schedule overruns or new requirements added during development. A high-priority requirement can be stated as “Those requirements that contribute the largest fraction of the total product value at the smallest fraction of the total cost”.

1.14 Categorization of Requirement Prioritization

Prioritization techniques can be divided into two categories as elucidated in the Fig 1.12:
1. Methods- Quantitative assigning values to different factors or criteria of requirements. The methods based category is further subdivided into two subcategories:

(a) Methods which process each requirement uniquely.

(b) Methods based on comparisons.

2. Negotiation approaches- The aim of negotiation approaches is to provide priority to requirements with the help of agreement that needs to be met between different stakeholders (Lehtola et al. 2004).

Determination of priorities in Negotiation approaches is based on discussions, negotiations among different stakeholders. Eg: Win-Win Model and Cumulative voting or 100 points.

1.15 Aspects of Requirement Prioritization

Berander and Andrews define an aspect as a property, factor, criteria or attribute that can be used to prioritize requirements. Requirements can be prioritized based on different aspects or criteria such as importance, time, volatility, cost, penalty, and risk.

1.15.1 Importance

In this aspect, the Stakeholders should prioritize which requirements are the most important for the systems performance. Importance is a multifaceted concept and it depends on the perspective which stakeholder is seeking for importance. Importance could be the urgency of implementation, an importance of requirement for product architecture and can also be strategic importance for the company, etc. It is necessary to specify
which kind of importance the stakeholders should prioritize in each case.

1.15.2 Penalty

The penalty is how much is needed to be paid if a requirement is not fulfilled or delivered on time. The penalty is a crucial aspect to be evaluated. Failing to meet those requirements that are having low values may cause a high penalty. For example, in case of failing to meet the standard imposed by on the software product, high penalty rate could be included even if the requirement is of low importance with respect to the customer. The penalty also occurs for implicit requirements failing to meet these requirements could make the product unsuitable for the market.

1.15.3 Cost

The organization estimates the cost for implementation of requirements. The following are the aspects on which cost depends:

- Reusability of the existing code.
- Requirements complexity.
- Testing and Documentation.

The main cost in software development depends on the number of hours spent. Therefore, Cost is expressed in terms of staff hours (effort) (Wiegers 1999).

1.15.4 Time

Time is calculated by aspects such as a successful implementation of the candidate requirement and by other major aspects such as parallelism degree in development, or staff training time etc (Wiegers 1999).

1.15.5 Risk

Risks are associated with each and every project. In project management, risk management is used to cope with both internal (technical and market risks) and external risks (eg regulation and suppliers). Based on the estimated risk and impact for each requirement, it is feasible to calculate the level of risk associated with the project. Risk management can also be used when devising requirements into products and releases by identifying risks that are prone to cause difficulties during development (Wiegers 1999). For example performance risks, process risks, schedule risks etc.
1.15.6 Other Aspects

Other aspects which affect requirement prioritization are the strategic benefit, volatility, available resources and market value. Requirements can be prioritized by the stakeholders depending on whether it is a single aspect or multiple aspects. Generally, prioritization of requirement is easier with respect to single aspect than on multiple aspects. For example, when a single aspect is considered such as market value, a requirement with a higher market value will have higher priority. Whereas, if consideration of more than one aspect is done such as cost with respect to market value, a higher market value requirement may generally cost high. So, stakeholders change their opinion, and the high priority requirement becomes the low priority requirement. Requirement prioritization aspects depend on one another and may interact with each other (for example, a high quality needs high cost etc.). Therefore change in one aspect results in a change in another aspect. Since the final product contains the influence of many aspects, so we have to consider multiple aspects in order to increase the extent of success in the final product.

1.15.7 Roles of Stakeholders in Requirement prioritization

Stakeholders or customers are a community of people who seek benefits from the projects in one or the other way. Major classification of customers involves:

- Specific customer.
- A group of customers.
- Mass-Market evolution.

The two extremes of software development are bespoke and mass-market development. In which systems using bespoke development

Bespoke development is used to develop a system with concern to a single customer which could further be used by many people e.g. an ATM machine is developed for a particular bank but all the clients of the bank make use of it whereas, in mass-market development, system is developed with a view that the whole world can make use of it. Though these two are the major extremes of the development there are many situations which exist among them, e.g. if we consider health sector and telecommunication sector, there exists many customers who make use of the software but neither of the two corresponds to any extreme of development.

Different prioritization mechanism is needed to be employed during different situations. Based on the criticality of the situation the prioritization techniques are chosen either single or a combination of methods are employed in order to finalize the set of
requirements due to the participation of various stakeholders. In situations where a single customer prioritizes requirements, it is much easier to perform prioritization because the requirements are prioritized in context to the customer alone. However, the end users are the one who is ultimately using the system (Berander 2004), so it is much advisable to prioritize requirements in context to end users too (Regnell et al. 2001). Whereas in situations where there are several customers prioritizing requirements it becomes difficult due of the involvement of numerous customers and their preferences, objectives, and viewpoints may vary diversely. Thus the main challenge is to arrive at win win situation where each stakeholder agrees to the requirements (Boehm et al. 2001)(Gruenbergacher 2000).

The Fig 1.13 elucidates the various stakeholders who are engaged with a project. The number of stakeholders participating in a project varies from one project to another. Stakeholders can be categorized as:

- Internal: Those stakeholders who are directly affected by the project.
- External: Those stakeholders who are indirectly affected by the project.

![Fig. 1.13 Stakeholders Catogarization](image)

1.16 Motivation

Software products preparation contains many phases in development process. The number of phases differ from one software model another. The modern world is dominating to prefer agile development model over other models, as it can handle frequently changing requirements. So while handling the changing requirements, we need to make sure
that we prioritize them. They are many existing methods for prioritizing the require-
ments. Which method is to be used in which scenario is to be known. They are different
methods saying about their own perspective of prioritization. As the agile development
emphasize for iterative development and releases, it is important to consider the cus-
tomer opinion while prioritizing. If this is considered, then it may raise interestingness
level in customer towards the organization. So considering this kind of factors may in-
crease the business value and satisfaction for the customer. The constraints that need to
be responded in this work are:

- Which algorithm sets the project?
- Which algorithm gives more advantages?

While considering these constraints, we need to consider many other constraints like the
complexity of the algorithm, ways of evaluating the performance of the algorithm. If we
are able to find better results in prioritization by using these algorithms than the existing
methods, then our work is productive. So to make the process of prioritization easier and
efficient we need to find some algorithms or methods which can handle a large number
of requirements efficiently. For this purpose, we can use some existing algorithms for
prioritization or we can refine or remodel the existing system as per the requirement of
the project or organization.

1.17 Objective

The objectives of this work are:

1. To propose a mathematical model this gives an accurate value for every require-
ment by considering different ranking methodologies.

2. To propose a method to prioritize the requirements by considering different peo-
ple opinion or importance over each requirement.

3. To propose an algorithm for prioritizing the requirements, which can also give
the dependencies between the prioritized requirements.

4. To propose a framework for agile development to prioritize the requirements de-
pending on both organizational and customer opinions.

1.18 Contribution

The main contributions of the work are:

- To study the factors affecting requirement prioritization using different ranking
methods and a linear model to predict the accurate importance of a requirement.
• Development of an algorithm for filtering the number of requirements by considering all stakeholders votes and opinions. Using a generic algorithm like the binary search tree to visualize the priorities in order.

• Discovery of frequently asked requirements using a data mining algorithm and finding the relation between the frequent requirements.

• Providing a framework which is revised from Scrum in agile development, to make it more compatible with requirement prioritization by considering customer conflict.

1.19 Organization of the thesis

The work is organized into seven chapters with a definite focus as listed:

Chapter 2 briefly defines a survey on prioritizing the requirements in software engineering process.

Chapter 3 Defines a framework which is used to filter the number of requirements using multi-voting among all stakeholders and binary search tree algorithms. This chapter concludes with the performance analysis of the proposed approach.

Chapter 4 Defines ranking methods that can be combined to a linear model, which can be said as an optimization based ranking approach for requirement prioritization WhaleRank. This chapter concludes with the performance analysis of the proposed approach.

Chapter 5 Defines a framework which describes the handling of stakeholder conflict by agile requirement prioritization using Apriori Technique. This chapter concludes with the performance analysis of the proposed approach.

Chapter 6 Defines a framework which remodels Scrum model for handling the prioritization of requirements based on stakeholder conflict in the agile mode of development. This chapter concludes with the performance analysis of the proposed approach.

Finally,

Chapter 7 Ends with conclusions and future work.