*Exploring Software Requirements*

#### Gangadhara Matti

#### Computer Software Engineering

#### Arizona State University, Mesa, USA

#### [gmatti1@asu.edu](mailto:gmatti1@asu.edu)

*Abstract*—Success or failure of the project depends on the requirements. Requirements Engineering is a creative problem solving which enables software engineers to apply various models, techniques and theories to understand and support the requirements phase in Software Development Life Cycle [1]. There are predefined steps that one to follow for effective and hazzle free transition of this step. This paper presents a roadmap for software requirements and existing tools that could be made use of.

Keywords—Software Requirements, Requirements elicitation, Requirements Validation.

# Introduction

Software requirements phase in Software Design Life Cycle is one of the important factors which drives the entire design process. This is the first step in any design process, be it Agile or waterfall or any other process [2]. While requirements Engineering is an old topic, an inability to frame problems and relate them to existing software development is one of the most glaring deficiencies.

This paper discusses about the strategy that one could follow to achieve better requirements. The SWEBOK [3] explains nicely the steps that one could take to achieve a smooth software requirements phase, this book is citated in many places of this paper.

# software requirements phase

Requirements Engineering is the term which is usually used to denote systematic steps which we need to follow for requirements gathering. Engineering as per Wikipedia is [4] “the use of scientific principles to design and build machines, structures and other things”. Thus, carrying out a blind requirement gathering would seriously hamper the software development process and in turn these bugs when carried forward in the later stages will cost a lot to fix. This in turn will delay the project being delivered late or would cost more a lot more than actual planned budget.

Paper [1] discusses the general approach that we can follow any requirements gathering phase. But we will stick to software requirements phase and this paper presents the detailed steps that one could follow to have a smooth transition from requirements to design phase and much further.

Section 1of this paper discusses the fundamental concepts that are required to understand the software requirements. Section 2 talks about the actual requirements process itself. Section 3 and 4 briefly introduces the software requirements elicitation and Analysis techniques that could be used for effective transition of these phases. Next section 5 and 6 talks about documenting these requirements which is nothing but Requirements Specification and validating these requirements. Finally concludes the paper by discussing the practical considerations in section 7 and available tools that could be made use of in section 8.

## Software Requirements Fundamentals

This section presents the terminologies, and concepts required to understand the software requirements phase. Understanding the terms which revolve around requirements engineering will help us understand the process better. This section presents the important terms that are prerequisites for better understanding the process.

## Software Reuirement Definition

According the definition of Wikipedia Software Requirements is defined as “a field within software engineering that deals with establishing the needs of stakeholders that are to be solved by software” [5]. So, the person collecting the requirements should be a consultant solving the problems of the customer, understanding their current workflow, routines to better solve the problem.

## Context Of Software Requirement

Understanding where the requirements phase fits in will give us an idea of the importance of this phase and this phase is like the basement of the project which you are building. Any mistakes that we make here will bubble up to the end and will have a huge impact on budget, time and every possible thing. Nishta Singh in paper [2] clearly describes the phases that we must follow for all Software Development life cycle. As per this paper in all the life cycle models, requirements definition is the first phase. Be it Agile or water fall or iterative or any other models.

## Functional and Non-Functional Requirements

In general, there are two broad categories of requirements functional and non-functional [6].

* Functional Requirements: These are the actual functions or tasks the user wants the application to do. For example, when you are designing a voice to text convertor, here one functional requirement would be the actual conversion of voice to text.
* Non-Functional Requirements: These are the properties of the application which are not directly related to user tasks. For example, the application should achieve for 5 nines of availability, should be consistent.

## Requirements Process

This phase briefly describes the process of requirements and how it fits in with the software engineering process [3].

* 1. *Process Models:* The requirements process is not a just a first step in Software Development Life Cycle. It is a process that will be carried out as the project grows, since the requirements are dynamic and keep changing.
  2. *Process Actors:* It is highly recommended to completely understand the different types of actors that will be interacting with the application. For example, for a banking application – there could be users, bank people fetching analytics, admin who will pitch in if something goes wrong and software engineers who will be working on the development of the software as well.
  3. *Process Quality and Improvement:* We should also measure the quality of requirements as we do for code or for any other things. Some things that could be considered are – requirements coverage, process benchmarking, security, planning and implementation.

## Requirements Elicitation

This is the first phase where in a Software Engineer start collecting the requirements from customer or stake holders. It is a human process where actors are identified and the relationship between these actors and the functionalities that are needed are identified and the solution to the problem starts evolving. It is highly important for the active involvement of the customer in this phase, as their requirements drive the project. The paper [7] clearly explains the step by step process for requirements elicitation for the development of web application, but this holds good for any general applications as well.

* 1. *Requirements Sources:* The sources for requirements are most important as they explain the project requirements in different angles. Below are few sources from which requirements can be gathered.
* Stakeholders and Users
* Documented papers
* Day-to-Day manual activities
  1. *Elicitation Techniques:* [8] Explains in depth the various techniques that could be followed for requirements elicitation that is collection the requirements. It also compares each of them. The important ones are mentioned below.
* Interviews – Interviewing the stakeholders or users is one the proved technique to gather the requirements in a neat and tidy way. It may be Structured, Unstructured or Semi-structured.
* Questionnaires – Requirements can be gathered via various set of questions and give some time for the respective person in charge to think and answer the questions.
* Domain Analysis – This could be done by reviewing the manual process in place or any documentation available to get the domain knowledge. However, this is application if the application is replacing any existing manual process or software.
* Prototyping – Prototyping the initial software with the requirements is proven to be the best method to get the feedback and clarify requirements with stakeholders. As this gives the actual feeling of working with the software.

## Requirements Analysis

Paper [9] presents the requirement analysis that could be done on a large, complex systems. But this is applicable in general to all systems as well. This is the phase where the software engineers analyze the requirements collected in previous phase.

* Analyze the requirements and connect them in a logical order. Resolve any conflicts that occur.
* Discover the boundary of the software use, where and how it will fit in the organization.
* Extract software requirements from system requirements too.
  1. *Requirement Clarification:* It is highly important to clarify all the ambiguous requirements. All of them must be crustal clear [9].
  2. *Conceptual Modeling*: Here we need to develop models which mimic real world. This process helps to understand the problem and arrive at the solution. There are several models which can be developed for depicting the project [3].
* Use Case Diagrams – This represents the user interaction with system and shows relationship between various use cases and users in the system.
* Data Flow Models – This depicts the overall data flow in the system.
* State Models – The state diagram of the project.
* Object Model – Modeling the various objects used in the project and their interactions.

Many of the above are a part of UML (Unified Modeling Language) [10]. Most of them are pictorial. These can be chosen based on scenarios for example if we are developing a system which has many objects in that case it is ideal to go with Object Model.

* 1. *Requirements Negotiation:* This is nothing but “Conflict Resolution”. If two stakeholders give conflicting requirements, the software engineer needs to resolve this and conclude.

## Requirement Specification

In this phase all the requirements are properly structured and well documented, and this document must be well reviewed, evaluated and approved [11]. This is specifically most important for a waterfall large project. As this will be the basis for overall development of the project. This large document can be split into 3 small document chunks, which are explained as below.

* 1. *System Definition Document:* User requirements document, records the system requirements.
  2. *System Requirements Specification:* Here all the system requirements are specified and documented. These are derived from software and non-software components.
  3. *Software Requirements Specification:* This establishes the agreement between the customers and the product development. All the features, how the system will work everything must be documented.

All these documents must be properly reviewed before finalizing. As this is the basis for the entire software to be developed [12].

## Requirements Validation

[13] All the requirements documents generated in the previous phases must be subject to validation and verification process. All these requirements must be complete, understandable, consistent and unambiguous. This also depicts the company standard.

Requirements Validation is a process of examining the requirements document to endure that it defines the right software [3]. Below are the steps that one could follow to have a systematic requirements validation workflow.

* 1. *Requirements Review:* The most straightforward means of evaluation of the correctness of the document is the manual review of the document. A group of people are assigned with the document of part of the document, they review the document and find out if there are any issues. It is advisable to have 4 eye or 6 eye approval, so to be extra sure of the correctness [13].
  2. *Prototyping:* This is commonly general accepted strategy for reviewing the correctness of the requirements by building a prototype and getting feedback from the customer on the same. Here mostly incomplete versions of the software is being developed and handed over to the customer for feedback. This facilitates increased user or customer involvement in the process even before its implementation. This greatly reduces time and cost as these bugs are identified at the earlier stage itself [13].
  3. *Model Validation:* It is atmost necessary to evaluate the correctness of the models developed during requirements phase. For example, for object models it is necessary to static analysis to verify the communication paths between different objects and its data flow [13].

## Practical Considerations

Not all companies follow this documentation process of requirements usually there will be change management which drive the project to be developed [3].

* 1. *Iterative Nature of the Requirements Process:* It is common for companies to follow iterative process of development life cycle such as agile, since the requirements in real-world keep changing. These models support for easy change of requirements and also addition anddeletion of requirements in the middle of the process as well.
  2. *Change Management:* [3]Nowadays due to the complexity of the project it is common for companies to use a tool to keep track of these requirements instead of hard document. The requirements in this tool guides the development of the software. Some examples of these tools are Jira, Taiga.
  3. *Measuring Requirements:* It is useful to measure the volume of the requirements; this gives the idea for budget and time considerations of the project.

## Software Requirements Tools

Due to the volatile nature of the requirements in these days, it is highly complex and challenging for manual requirement management [14]. To overcome such problems, there are tools which facilitate collection and management of software requirements. Below are the few important tools that are used by big companies to manage requirements. This is applicable to both iterative and waterfall models [14].

* 1. *MaramaAI:* Automated Inconsistency Checker tool for requirement quality improvement.
  2. *ARM:* Reconstructed Automated Requirement Measurement for requirement specification.
  3. *C and L:* Performs requirement description transformation into UML diagrams.
  4. *Ontology:* Interaction diagrams, information sets and free-text information.
  5. *Cradle:* This tool provides different tools including requirement capturing, software requirement management. This maximizes security, efficiency and traceability of requirements.

# Conclusion

There is a long-standing recognition of Software Requirements phase inability to properly gather the requirements and map it to the existing problem to be solved. Understanding the existing Requirements Engineering and how fit in together would help in developing a framework which suits the problem to be solved. This also helps software engineers to choose the best possible approaches to solve their problems rather than reinventing the wheel from scratch.

# ACKNOWLEDGEMENT

I would like to thank Professor Michael Findler for assigning me this interesting paperwork. I got to learn a lot by reviewing other papers. He did help us in gaining a lot of insights in Software Requirements. Additionally, I would like to thank out TA Jahnvi Rai for helping us all through the start to finish of this work.

# Bibliography

|  |  |
| --- | --- |
| [1] | S. M. B. P. C. C. V. Alistair Mavin, "Towards an Ontology of Requirements Engineering approaches," in *IEEE 27th International Requirements Engineering Conference (RE)*, UK, 2019. |
| [2] | N. Singh, "Agile Software Development: An Overview," in *IJRCEMAS*, 2016. |
| [3] | P. B. a. R. E. Fairley, SWEBOK - The guide to Software Engineering Body of Knowledge, IEEE, 2014. |
| [4] | "WIKIPEDIA - THE FREE ENCYCLOPEDIA," WIKIPEDIA , 19 02 2020. [Online]. Available: https://en.wikipedia.org/wiki/Software\_requirements. [Accessed 26 04 2020]. |
| [5] | "Wikipedia, the free encyclopedia," en.wikipedia.org, 19 02 2020. [Online]. Available: https://en.wikipedia.org/wiki/Software\_requirements. [Accessed 26 04 2020]. |
| [6] | F. Dalpiaz, "Agile Requirements Engineering with User Stories," in *2018 IEEE 26th International Requirements Engineering Conference*, Netherlands, 2018. |
| [7] | M. Ghasemi, "What Requirements Engineering can Learn from Process Mining," in *2018 1st International Workshop on Learning from other Disciplines for Requirements Engineering*, Canada, 2018. |
| [8] | P. S. Alistair Sutcliffe, "Requirements Elicitation: Towards the Unknown Unknowns," in *IEEE*, Lancaster, 2013. |
| [9] | M. B. a. G. Mark, "Rwquirements Analysis: A Deeper Examination," in *IEEEE*, California, USA, 2003. |
| [10] | B. Westphal, "Teaching Software Modelling in an Undergraduate Introduction to Software Engineering," in *ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion*, Freiburg, Germany, 2019. |
| [11] | E. K. a. C. E. Boustani, "Assessing the Quality of Software Requirements Specifications," in *16th IEEE International Requirements Engineering Conference*, Hannover, Germany. |
| [12] | K. W. a. M. Smiałek, "Reusing Terminology for Requirements Specifications from WordNet," in *16th IEEE International Requirements Engineering Conference*, Hamburg, 2008. |
| [13] | W. K. X. L. Yilong Yang∗, "RM2PT: Requirements Validation through Automatic Prototyping," in *2019 IEEE 27th International Requirements Engineering Conference (RE)*, Macau, 2019. |
| [14] | M. A. A. F. S. a. J. J. A. B. Atif Shah, "An Evaluation of Software Requirements Tools," in *The 8th IEEE International Conference on Intelligent Computing and Information Systems (ICICIS 2017)*, Dhahran, Saudi Arabia, 2017. |