An Effective Requirement Engineering Process Model for Software Development and Requirements Management

Dhirendra Pandey
Department of Information Technology
Baba Saheb Bhimrao Ambedkar University
Lucknow (UP), India
e-mail: prof.dhiren@gmail.com

U. Suman
School for Information Science & IT
Devi AhilyaVishwavidyalaya
Indore (MP), India
e-mail: ugrasen123@yahoo.com

A. K. Ramani
School for Information Science & IT
Devi AhilyaVishwavidyalaya
Indore (MP), India
e.mail: ramaniak@yahoo.com

***Abstract*- Requirement engineering is the most effective phase of software development process. It aims to collect good requirements from stakeholders in the right way. It is important for every organization to develop quality software products that can satisfy user's needs. Requirements engineering for software development process is a complex exercise that considers product demands from a vast number of viewpoints, roles, responsibilities, and objectives. Therefore, it becomes necessary to apply requirement engineering practices in every phase of software development process. In this paper, we propose an effective requirements engineering process model to produce quality requirements for software development. Requirement management and planning phase is executed independently for an effective management of requirements. It is iterative in nature for better requirement engineering and later maintenance. The successful implementation of proposed requirement engineering process can have a good impact on the production of quality software product.**

***Keywords:* Requirement engineering, requirement analysis, requirement elicitation, requirement development, requirement management.**

I. INTRODUCTION

Requirements are attributes or something, which we discover before building products. It is a condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents [1]. A well-formed requirement is a statement of system functionality that satisfies customer needs. There exists a reciprocal interrelationship between human beings and machines for requirement gathering that can assist to produce quality products [2]. Requirements are commonly classified as functional and non-functional [1]. A functional requirement is a requirement that specifies an action performed by a system without considering physical constraints. Non-functional requirement specifies system properties such as environmental and implementation constraints, performance, platform dependencies, maintainability, extensibility, reliability etc. [1].

Requirement engineering is generally accepted to be the most critical and complex process within the development of socio-technical systems [3, 4, 5]. In this way, it helps to describe a multidisciplinary role of requirements engineering process as well as the patterns for social interaction. The main reason is that the requirements engineering process has the most dominant impact on the capabilities of the resulting product. Furthermore, requirements engineering is a process in which most diverse set of product demands from the most diverse set of stakeholders, which is already being considered by the practitioners. These two reasons make requirements engineering complex as well as critical.

Requirement engineering is a systematic approach through which the software engineer collects requirements from different sources and implements them into the software development processes. Requirement engineering activities cover the entire system and software development life cycle. Requirements engineering process is an iterative process which also indicates that the requirements management is understood as an aspect of requirements engineering process [9, 10, 11]. Traditionally, requirements engineering is performed in the beginning of the system development lifecycle [13]. However, in large and complex systems development, developing an accurate set of requirements that would remain stable throughout the months or years of development has been realised to be impossible in practice [14]. Therefore, requirements engineering is an incremental and iterative process, performed in parallel with other system development activities such as design, coding etc.

Requirements engineering contains a set of activities for discovering, analysing, documenting, validating and maintaining a set of requirements for a system [6]. Requirements engineering is divided into two main groups of

978-0-7695-4201-0/10 $26.00 © 2010 Crown Copyright
DOI 10.1109/ARTCom.2010.24

activities; namely, requirements development and requirement management. Requirement development covers activities related to discovering, analysing, documenting and validating requirements where as requirement management includes activities related to traceability and change management of requirements. Requirements verification consists of those activities that confirm that the product of a system development process meets its technical specifications. Requirements validation consists of activities that confirm that the behaviour of a developed system meets its user needs [7].

Requirement engineering is a very important activity, which can affect the entire activity of software development project. Requirement engineering is one of the most important tools for gathering requirements, which is concerned with analysing and documenting the requirements [8]. We propose an effective model of requirement engineering process for software development, which is discussed in detail with various phases in Section 2. The comparative discussion of proposed requirement engineering process model with existing models is presented in Section 3. Finally, Section 4 describes the concluding remarks and future research work.

II. REQUIREMENT ENGINEERING PROCESS

The main objective of requirement engineering is to discover quality requirements that can be implemented into software development. The identified requirements must be clear, consistent, modifiable and traceable to produce a quality product. In this paper, we have proposed an effective requirement engineering process model, which is shown in figure 1. It consists of mainly four phases, namely; requirement elicitation and development, documentation of requirements, validation and verification of requirements, and requirement management and planning. Requirement elicitation and development phase includes requirement analysis and allocation and flow down of requirements.

Documentation of requirements includes identification of requirements and software and system requirement specification. Validation and verification of requirements phase is concerned with conforming the documented requirements.

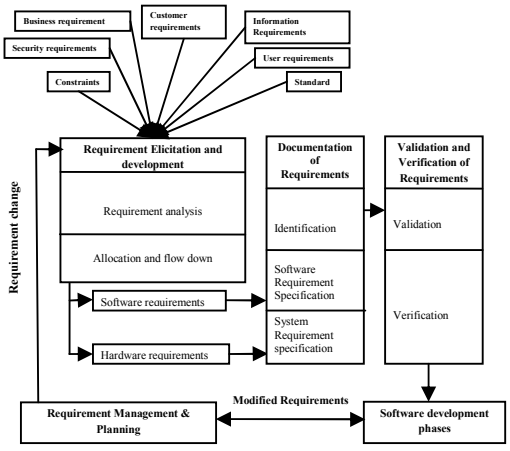

Figure. 1: Requirement Engineering Process Model

The requirement management and planning phase controls the continuously changing requirements. Each of these activities is further detailed in the following sub sections. The proposed process describes requirements engineering for software development systems in which requirement engineering must be a part of the software development process.

A. Requirements Elicitation and Development

Requirement elicitation and development phase mainly focuses on examining and gathering desired requirements and objectives for the system from different viewpoints (e.g., customer, users, constraints, system's operating environment, trade, marketing and standard etc.). Requirements elicitation phase begins with identifying stakeholders of the system and collecting raw requirements from various viewpoints. Raw requirements are requirements that have not been analysed and have not yet been written down in a well-formed requirement notation. The elicitation phase aims to collect different viewpoints such as business requirements, customer requirements, user requirements, constraints, security requirements, information requirements, standards etc. Typically, the specification of system requirements starts with observing and interviewing people [15].

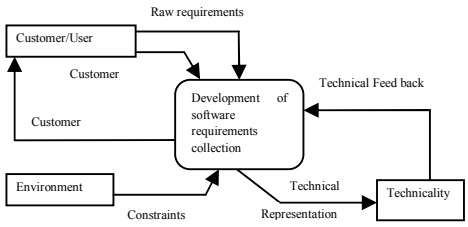

Figure. 2: Development of Requirements

Furthermore, user requirements are often misunderstood because the system analyst may misinterpret the user's needs. In addition to requirements gathering, standards and constraints play an important role in systems development. The development of requirements may be contextual, which is shown in figure 2. It is observed that requirement engineering is a process of collecting requirements from customer and environment in a systematic manner. The system analyst collects raw requirements and then performs detailed analysis and receives feedbacks. Thereafter, these outcomes are compared with the technicality of the system and produce the good and necessary requirements for software development [3, 35].

1) *Requirements Analysis:* The development and gathering of good quality requirements is the basic activity of any organisation to develop quality software products. These requirements are then rigorously analysed within the context of business requirements. It is also observed that the identified raw requirements may be conflicting [29]. Therefore, negotiation, agreement, communication and prioritisation of the raw requirements become important activities of requirement analysis. The analysed requirements need to be

documented to enable communication with stakeholders and future maintenance of requirements and the system. Requirements analysis also refines the software allocation and builds models of the process, data, and behavioural domains that may be treated by the software. Prioritising the software requirements is also part of software requirements analysis.

2) *Allocation and Flow-down of Requirements:* The purpose of requirements allocation and flow-down is to make sure that all system requirements are fulfilled by a subsystem or by a set of subsystems that can work together to achieve objectives. Top-level system requirements need to be organized hierarchically that can assist to view and manage information at different levels of abstraction. The requirements are decomposed down to the level at which the requirement can be designed and tested. Thus, allocation and flow-down may be performed for several hierarchical levels. The level of detail increases as the work proceeds down in the hierarchy. That is, system-level requirements are general in nature, while requirements at low levels in the hierarchy are very specific [17, 7]. The top-level system requirements defined in the system requirements development phase are the main input for the requirements allocation and flow-down phase. As the system level requirements are being developed, the elements that should be defined in the hierarchy should also be considered [14]. Allocation and flow- down of requirements include:

a) *Allocation of Requirements:* Allocation of requirements is an architectural task carried out in order to design the structure of the system and to issue the top-level system requirements to subsystems. Architectural models provide the context for defining interaction between applications and subsystems to meet the requirements of the system. The goal of architectural modelling is to define a robust framework within which applications and component subsystems may be developed [15]. Each system level requirement is allocated to one or more elements at the next level. Allocation also includes allocating the non-functional requirements to system elements. Each system element will need an apportionment of non-functional requirements (e.g., performance requirement) [14, 18, 19, 10]. When functional and the non-functional requirements of the system have been allocated then the system engineer can create a model that represents the interrelationship between system elements and sets a foundation for later requirements analysis and design steps.

b) *Flow-down of Requirements:* Flow-down consists of writing requirements for the lower level elements in response to the allocation. When a system requirement is allocated to a subsystem, the subsystem must have at least one requirement that responds to the allocation. The lower-level requirements either may closely resemble to the higher level or may be very different if the system engineers recognize a capability that the lower level element must have to meet the higher-level requirements. The lower-level requirements are often referred to as derived requirements [14]. Derived requirements are requirements that must be imposed on the subsystem(s). These requirements are derived from the systems decomposition process. There are two subclasses of derived requirement, i.e. subsystem requirements and interface requirement. The subsystem requirements are the requirements that must be imposed on the subsystems themselves but do not necessarily provide a direct benefit to the end user. Interface requirements are the requirement that arise when the subsystems need to communicate with one another to accomplish an overall result. This result is needed to share data or power or a useful computing algorithm. [17].

In the allocation and flow-down phase, requirements identification and traceability have to be ensured both to higher level requirements as well as between requirements on the same level. The rationale behind design decisions should be recorded in order to ensure that there is enough information for verification and validation of the next phases work products and change management. In theory, it produces a system in which all elements are completely balanced or optimized. In the real world, complete balance is seldom achieved, due to fiscal, schedule, and technological constraints [11, 18]. Allocation and flow-down starts as a multi-disciplinary activity, i.e., subsystems may contain hardware, software, and mechanics. Initially, they are considered as one subsystem but different disciplines are considered separately in later iterations.

B. Documentation of Requirements

A formal document is prepared after collecting requirements, which contains a complete description of the external behaviour of the software system. Requirements development process is the activity of determining which functionality of the system will be performed by software. Non-functional requirements are combined together with functional requirements into the software requirements specification with the help of flow-down, allocation, and derivation. A software requirements specification will be established for each software subsystem, software configuration item, or component is part of this phase [11]. The documentation of requirements includes requirement identification and requirement specification.

1) *Requirements Identification:* Requirements identification practices focus on the assignment of a unique identifier for each requirement [6]. These unique identifiers are used to refer requirements during product development and management. Requirements identification process consists of three sub activities. The basic numbering activity includes significant numbering and non-significant numbering whereas identification activity includes labelling, structure based identification and symbolic identification. The last technique is to support and automate the management of items, which includes dynamic renumbering, database record identification and base lining requirements [6, 22].

2) *Requirement Specification:* Requirements specification document is produced after the successful identification of requirements. The document describes the

product to be delivered rather than the process of its development. Software requirement specification (SRS) is an effective tool for requirement specification which is a complete description of the behaviour of the system or software to be developed. It includes a set of use cases that describe all the interactions that users will have with the system/software [12]. In addition to use cases, the SRS also contains non-functional (or supplementary) requirements. Non-functional requirements are requirements which impose constraints on the design or implementation. SRS is a comprehensive description of the intended purpose and environment for software under development. The SRS fully describes what the software will do and how it will be expected to perform. An SRS minimizes the time and effort required by developers to achieve desired goals and also minimizes the development cost. A good SRS defines how an application will interact with system hardware, other programs and users in a wide variety of real-world situations. Parameters such as operating speed, response time, availability, portability, maintainability, footprint, security and speed of recovery from adverse events are evaluated in SRS.

C. *Requirements Verification and Validation*

When the entire requirement are described and specified in the SRS, then different parties involved have to agree upon its nature. One should ascertain that the correct requirements are stated (validation) and these requirements are stated correctly (verification). Validation and verification activities include validating the system requirements against raw requirements and verifying the correctness of system requirement documentation. The most common techniques for validating requirements are requirements reviews with the stakeholders, and prototyping. Software requirements need to be validated against system level requirements and SRS needs to be verified. Verification of SRS includes correctness, consistency, unambiguousness and understandability of requirements. In requirement verification and validation, requirements traceability mechanism can generate an audit trail between the software requirements and finally tested code. Traceability should be maintained to system level requirements, between software requirements, and to later phases, e.g., architectural work products. The outcome of the software requirements development phase is a formal document including a baseline of the agreed software requirements [28].

D. *Requirement Management and Planning*

Requirements Management and planning phase controls and tracks the changes of agreed requirements, relationships between requirements, and dependencies between the requirements documents and other documents produced during the systems and software engineering process [9]. It is a continuous and cross-section process that begins from requirements management planning and continues activities of identification and change control during and after requirements development process phases. Requirements management is a continuous activity that can perform after development and during maintenance because requirements may continue to change [9, 21].

Requirement traceability is a part of requirement management, which refers ability to describe and follow the life of a requirement and its relations with other development artefacts in both forwards and backwards direction [22, 23]. We need to manage continually changing requirements because these are often changed time to time and cause of excessive damage. The changing requirements can be managed by applying requirements change management process. It refers to the ability to manage changes to requirements throughout software development lifecycle. Change management includes identifying, analysing, deciding on whether a change will be implemented and the possibility of implementing and validating change requests.

Requirement management is sometimes considered as the most complex requirements engineering process [24]. A requirement change can have a large impact on the developing system, which is very hard to estimate. For every change, costs and re-development work must have to be considered before approving the change. After refinement of requirements, the requirement can be further incorporated into software development processes. Requirement change management has a strong relationship with baselining [25]. Requirements management tools have been developed to manage unstable requirements and large amount of data collected during requirements engineering process [16].

Requirements management tools collect together with the system requirements in a database or repository and provide a range of facilities to access the information about the requirements [9]. Software requirements management tool support secure and concurrent co-operative work between members of a multidisciplinary development team that support standard systems, modelling techniques and notations. It is necessary to maintain an audit trail of changes, archive baseline versions, and engage a mechanism to authenticate and approve change requests. One should also maintain a comprehensive data dictionary of all project components and requirements in a shared repository and produce predefined and ad-hoc reports. Documents should also be generated that can comply with standard industrial templates [26].

III. DISCUSSION

The proposed requirement engineering process is more effective to produce quality requirements. The other requirement engineering processes are limited to cover only few dimension of requirement engineering such as requirement elicitation, requirement specification and requirement verification and validation [10]. The proposed model introduces all important and hidden aspects of requirement engineering process. The existing requirement engineering process models are unable to communicate their phases with software development process in the right manner [10, 30]. We relate all the important aspects of requirement engineering process to software development process in order to find out good requirements from various sources that can be implemented into software development process for producing

quality software products. We have also relate requirement management and planning phase to the software development phases in our model because requirements can change over the time and during software development process, this can give bad outcomes. Therefore, it is necessary to manage continuously changing requirements through requirement management and planning.

IV. FUTURE SCOPE

The proposed requirement engineering model can be used in larger software development process, where the requirements are continuously changed. It presents the new insight of requirement management and planning which can manage the changing requirement. The requirement engineering activities in the model such as elicitation, documentation of requirement and verification & validation are tightly interconnected to the software development phases. It can help to recover and modify the requirement, whenever requirements are changed in any phase of software development. Software developer can easily supply the required and modified requirements using requirement management and planning activity. Using this model, software developer can design the software product, which will be able to avoid and manage changing requirements.

The proposed model is iterative in nature, which is useful in requirement prototyping with more user participation. Using this model, organisation can allow users for any further change in requirements during software development process. The SRS can be modified accordingly and these changed requirements can be re-implemented in software development.

V. CONCLUSION

Requirements engineering is the initial phase of software engineering process in which user requirements are collected, understood, and specified for developing quality software products. The requirement engineering process deserves a stronger attention in the industrial practices [27]. In this paper, we proposed an effective requirement engineering process model for software development that can be used for software development processes to produce a quality product.

REFERENCES

[1]. P. Jalote, An Integrated Approach to Software Engineering, 3rd edition, Narosa Publishing house, India, 2005.

[2]. G. Ropohl, "Philosophy of Socio-technical Systems", In Society for Philosophy and Technology, 1999, pp. 59-71.

[3]. Pandey, U. Suman, A. K. Ramani, "Social-Organizational Participation difficulties in Requirement Engineering Process- A Study", National Conference on Emerging Trends in Software Engineering and Information Technology, Gwalior Engineering College, Gwalior,2009.

[4]. N. Juristo, A. M. Moreno & A. Silva, "Is the European Industry Moving Toward Solving Requirements Engineering Problems?" IEEE Software, 2002, pp. 70-77.

[5]. S. Komi-Sirvio & M. Tihinen, "Great Challenges and Opportunities of Distributed Software Development - An Industrial Survey", Fifteenth International Conference on Software Engineering and Knowledge Engineering, SEKE2003, San Francisco, 1-3 July 2003.

[6]. J. Siddiqi, "Requirement Engineering: The Emerging Wisdom", IEEE Software, 1996, pp.15- 19.

[7]. Sommerville & P. Sawyer, Requirements Engineering: A Good Practise Guide. John Wiley & Sons, 1997.

[8]. D. Pandey, U. Suman, A. K. Ramani, "Impact of Requirement Engineering Practices in Software Development Processes for Designing Quality Software Products", National Conference on NCAFIS, DAVV, Indore, 2008.

[9]. R. Stevens, P. Brook, K. Jackson & S. Arnold, Systems Engineering - Coping with Complexity, Prentice Hall, London, 1998.

[10]. G. Kotonya & I. Sommerville, Requirements Engineering: Process and Techniques. John Wiley & Sons, 1998.

[11]. J. D. Sailor, System Engineering: An Introduction. IEEE System and Software Requirements Engineering, IEEE Software Computer Society Press Tutorial. IEEE Software Society Press, 1990.

[12]. Pandey, U. Suman, A. K. Ramani, "Design and Development of Requirements Specification Documents for Making Quality Software Products" National Conference on ICIS, D.P. Vipra College, Bilaspur, 2009.

[13]. R. H. Thayer & W. W. Royce, Software Systems Engineering, IEEE System and Software Requirements Engineering. IEEE Software Computer Society Press Tutorial. IEEE Software Society Press. Los Alamos, California, 1990.

[14]. W. W. Royce, "Managing the Development of Large Software Systems" Proceedings of IEEE Wescon, Reprinted in Proceedings 9th International Conference Software Engineering IEEE Computer Society Press, Los Alamitos. California. USA, 1987, pp. 328-338.

[15]. M. Dorfman, System and Software Requirements Engineering IEEE System and Software Requirements Engineering, IEEE Software Computer Society Press Tutorial. IEEE Software Society Press. Los Alamos, California, 1990.

[16]. S. W. Ambler, Process Patterns. Building Large-Scale Systems Using Object Technology. Cambridge University Press, 1998.

[17]. P. Parviainen, H. Hulkko, J. Kaariainen, J. Takalo & M. Tihinen, "Requirements Engineering", Inventory of Technologies, VTT Publications, Espoo, 2003.

[18]. Leffingwell & D. Widrig, Managing Software Requirements - A Unified Approach, Addison-Wesley, 2003.

[19]. D. Nelsen, System Engineering and Requirement Allocation, IEEE System and Software Requirements Engineering, IEEE Software Computer Society Press Tutorial, IEEE Software Society Press Los Alamos, California, 1990.

[20]. R. S. Pressman, Software Engineering, A Practitioner's Approach. Third Edition, McGraw- Hill Inc, 1992.

[21]. S. Blanchard & W. J. Fabrycky, Systems Engineering and Analysis. Prentice-Hall, 1981.

[22]. S. Lauesen, Software Requirements: Styles and Techniques, Addison-Wesley, 2002.

[23]. H. Berlack, Software configuration management. John Wiley & Sons, 1992.

[24]. O. Gotel, Contribution Structures for Requirements Traceability. Ph.D. Thesis, Imperial College of Science, Technology and Medicine, University of London, 1995.

[25]. M.E.C Hull, K. Jackson & A. J. J Dick, Requirements Engineering. Springer-Verlag. Berlin, 2002.

[26]. Hooks & K. Farry, Customer-Centred Products, Amacom, 2001.

[27]. M. Lang & J. Duggan, "A Tool to Support Collaborative Software Requirements Management" Requirements Engineering Journal 6(3), 2001, pp. 161–172.

[28]. T. Gilb, Competitive Engineering. Addison-Wesley, 2003.

[29]. Paivi Parviainen, Maarit Tihinen, Marco Lormans & Rini van Solingen, Requirements Engineering: dealing with the Complexity of Sociotechnical Systems Development, 1998.

[30]. L. A. Macaulay, *Requirements Engineering*, Springer-Verlag, 1996.