Comparison of various Elicitation Techniques and Requirement Prioritisation Techniques

Nilofar Mulla
Department of Information Technology, MIT Pune 38, Maharashtra, India

Sheetal Girase
Assistant Professor, Department of Information Technology, MIT Pune 38, Maharashtra, India

Abstract

Software engineering research has been, and still is criticised as being immature and unscientific due to lack of evaluation. However, software engineering community is now focusing more on empirical research. One of the major activities within the requirements engineering process is to use requirements elicitation and requirements prioritization that helps to focus on the most important requirements. Requirement elicitation is recognized as one of the most critical knowledge intensive activities of the development of software. Requirements prioritization aims at identifying the most important requirements for a system .There are many elicitation and prioritization techniques available; still there is lack of evidence of which technique to prefer. The reasons could be the differences in contexts, measurement of variables and usage of data sets. In this paper, the area of requirements elicitation and prioritization has been systematically reviewed in order to assess what evidence regarding different techniques exist.

1. Introduction

"Software engineering does not yet have a widely recognized and widely appreciated set of research paradigms in the way that other parts of computer science do"[4]. That is, we don't recognize what our research strategies are and how they establish their results. Shaw, in her article says "described as a challenge to the whole of software engineering community" [8]. The main challenge of the software engineering community is to satisfy the customer needs and possibly exceed his expectations in an economic, rapid and profitable manner. The process of Requirements engineering has the potential of fulfilling the customer needs and possibly exceeding them by using rigorous and defined methodologies. Requirements engineering can help organizations develop quality software systems within time and budget constraints which are true reflection of customer needs [4]. One of the major activities within requirements engineering process is to use requirements prioritization that evaluates requirements to focus on the most important ones. The primary success factor of requirements elicitation is that requirements meet end user needs. This outcome is difficult to achieve because users often have trouble identifying and articulating their needs and because those needs often change as a result of system implementation. This difficulty is compounded for newer technologies such as data warehouses because requirements continue to evolve over time as users become familiar with the systems and their needs for information change. For these technologies, system requirements are a moving target. Over time, challenges arise from the simultaneous evolution of the technology and of the users' requirements. For these reasons, calls for effective user involvement in requirements elicitation continue. Effective requirements elicitation depends upon the ability of users and analysts to understand and appreciate one another's words. This represents a significant, but not insurmountable, challenge which we explore in this paper. Requirements elicitation is an often poorly completed aspect of systems analysis. Mistakes made in elicitation have been shown many times to be major causes of systems failure or abandonment and this has a very large cost either in the complete loss or the expense of fixing mistakes.

2. Challenges in Requirements Elicitation

Domain experts, customers, and users are essential during requirements elicitation; however, they do not necessarily understand the intricacies of software development. On the other hand, software engineers are likely to be unfamiliar with the application domain. This creates a communications barrier between the software engineers and the domain experts, customers, and users, which can be overcome by formal elicitation methods. Requirements elicitation involves end users and analysts interacting to identify and 'capture' the data and processes that will make up the eventual system. User-analyst communication is an important part of requirements elicitation, but communication styles and techniques most readily associated with requirements elicitation - interviews and questionnaires, for instance - arc rarely sufficient to elicit the whole range of requirements [6]. The use of such standard 'instruments' in any user-analyst exchange introduces the potential for errors of omission that arise as a consequence of the

analyst's difficulty in eliciting system requirements that are outside the instrument's scope. The majority of requirements elicitation techniques fail to address the less conspicuous and often more tacit requirements, priorities, and issues that analysts do not know to ask about and those users do not or cannot readily identify and articulate. Traditional techniques are unable to fully diagnose how such contextual issues will affect system requirements, system development, and system evolution. Furthermore, analysts need unbiased, systematic approaches during communication to assist users in identifying and articulating needs. To overcome the limitations and perceptual biases of traditional requirements elicitation approaches, the concept of user-centred analysis - the process of 'capturing' requirements from the user's point of view - has frequently been promoted as a means to achieve a more comprehensive understanding of end user system needs.

3. Requirements Elicitation

Developing a large system is a complex and difficult process. In the early days of computing, there was no particular organisation to this process: programmers just sat down and tried to write code that would be useful. To- day, few doubt that a task that can consume hundreds of person-years should be carefully planned and managed. Therefore the system \life cycle" has been broken into a number of so called \phases," of which Requirements Engineering is the earliest phase3 that lies largely within Computing Science. The requirements phase is typically preceded by business planning, and is formally initiated by the client.

Requirements describe goals, functions, and constraints of a software system. The term “elicitation” is preferred to “capture”, to avoid the suggestion that requirements are out there to be collected simply by asking the right questions [5]. Rather, the data collected during requirements elicitation often has to be interpreted, analyzed, modelled, and validated. “Elicitation” is also referred to as “acquisition” in some literature.

3.1 Requirements Elicitation techniques

Following is the table (Table 1) showing various Requirements Elicitation techniques [7] along with their advantages and disadvantages.

There are a variety of techniques that can be employed to elicit requirements. The approach taken by a requirements engineer is not limited to one particular technique. Organizational processes, application type, available resources, and individual preference all play a role in determining a particular approach. For instance, applications that need early customer feedback might benefit from the use of prototyping combined with group elicitation. The requirements elicitation process involves all stakeholders, which includes customers, developers, and users. Better technique selection will improve the quality of the requirements elicitation process and increase the success of software development projects.

4. Requirements Prioritisation

After requirements are identified, they also need to be prioritised. Requirement prioritization process is used to determine which candidate requirement of a software project should be included in a certain release, for this purpose different techniques are used [1]. Projects often have more requirements than time, resource, and budget allow for. A function can always be added and the user interface enhanced. Some requirements are critical for the success of the software system. Hence, requirements should be prioritised so that the ones that are most likely to achieve customer satisfaction can be selected for implementation. It is essential to decide what is important before these requirements are incorporated into the software development process. By addressing the high-priority requirements before considering the low-priority ones, one can significantly reduce both the costs and duration of a project.

4.1 Requirements Prioritisation techniques

Following is the table (Table 2) showing various Requirements Prioritization techniques [2] along with their advantages and disadvantages.

Requirement Prioritization is the most important step in requirement engineering process. Without assigning proper and accurate requirement to each release, it is almost impossible to complete project on time and within budget. In a review of the state of the practice in requirements engineering, Lubars found that many organizations believe that it is important to assign priorities to requirements and to make decisions about them according to rational, quantitative data. Still it appeared that no company really knew how to assign priorities or how to communicate these priorities effectively to project members. According to us all the factors listed in (Table 2) should be considered while prioritizing requirements. It is better to spend time in choosing the right requirements for releases rather than choosing the wrong ones and wasting time, budget and resources. Before starting prioritizing requirements first of all check the dependencies between requirements (Dependency Constraints). If dependencies between requirements are not taken properly it gets very difficult to select an appropriate requirement sets for the releases, because it is highly possible that you are selecting a requirement in the current release and leaving another for the next release but both requirements should be in the same release. Therefore, the first step is to check dependency constraints between requirements. It is almost impossible to implement all the requirements in one release. That is why some sort of prioritization process is needed to implement the most important requirements in the first release and leave the less important ones for the future releases.

5. Conclusion

Requirements engineering interacts with many technical as well as social sciences to improve the process of eliciting, analyzing, documenting and maintaining requirements. Requirements elicitation is an often poorly completed aspect of systems analysis. Mistakes made in elicitation have been shown many times to be major causes of systems failure or abandonment. This has a very large impact on the cost either in terms of complete loss or the expenses on fixing mistakes. In this paper, the different requirement elicitation methods are studied, compared and discussed. Requirements prioritization is highly emphasized area within requirements engineering that helps different stakeholders decide on the final set of requirements, which will eventually make up a system. Certain techniques exist within requirements prioritization with their own advantages and limitations. All these techniques could be of valuable help for organizations to decide which requirements are important and which are less important in the overall development of a project. Requirements elicitation is a critical step in the requirements development process. It is consequently imperative that requirements engineers apply appropriate methods to perform the process sufficiently. This paper has attempted to present meaningful insights into the feature of different types of requirements elicitation techniques.

Table 1. Requirement Elicitation Techniques

Method	Description	Advantages	Disadvantages
Interviews	Analyst discusses the desired product with different groups of people and builds up an understanding of their requirements. If the interview is conducted with pre-defined agenda and questions, it is called structured interview; otherwise, it is an open-ended interview.	• Collecting the rich and detailed data • Collecting information to design a survey or other usability activity • Getting a holistic view of the whole system	• Collecting data from large samples or people • When it need to collect the data very rapidly
Workshop, focus groups	Stakeholder representatives gather together for a short but intensely focused period to create or review high -level features of the desired products.	• This technique is very much effective to resolve the conflicts among customers in order to bring them at one table. • Each and every aspect of requirements is discussed and proper suggestions are given using group work. • The stakeholders provide the direct remarks about the software requirements. • Stakeholders work in the environment. • Group work Provides the remarkable	• This technique needs a lot of effort as compared the other requirements engineering techniques. • Sometimes all the stakeholders can join at the same time as it may be possible that they may be busy in other tasks. • Group work is less effective in the highly political tense situation.
Brainstorming	Stakeholder representatives gather together and rapidly develop a large and broad list of ideas. It encourages "out -of-the-box" thinking without normal constraints, and involves both idea generation and idea reduction.	• Brainstorming is mostly used for the innovative sort of projects where each participant provides his or her own ideas after their personal research about the project to be started. • This technique is often used make the key decisions about the requirements of the project. • It promotes free thinking and expression of ideas. • Brainstorming provides the innovative ideas about the project to be developed.	• Brain storming is seriously affected by exploring the critique ideas. • Brainstorming is not used to resolve the major issues.
Scenarios, passive Storyboards	It is an interaction session to describe a sequence of actions and events for a specific case of some generic task which the system is intended to accomplish. Clarified system requirements related to procedures and data flows of a task. In a highly uncertain situation, an effective and relatively inexpensive way to develop an initial set of requirements.	Because storyboards exist independently of the software system they describe, they have many advantages over regular prototypes. They cannot crash, are very easy to share with large groups, and do not give the false impression that the system is already developed. Additionally, feedback is easier to accommodate.	One of the biggest problems with storyboards is that they can become outdated very quickly. User interfaces originally defined often change over time, and that creates a maintenance burden.

Prototyping	Prototype is a version of a product launched into market to provide the so for services to the customers. Prototyping is used to provide a version of the software and which is not final so that the customer can gain the experience and also may be able to provide other requirements that need to be implemented in the next prototyping. The response of the user is in the form of a feedback which is recorded as like requirements of the system.	• Prototyping provides the detail information by investing each and every prototype by the customer. • Prototypes are mostly used in conjunction with other elicitation techniques such as interviews. • Prototypes useful when developing human computer GUI interfaces. • Prototypes provide a good chance to the stakeholders an effective rule and to be involved in the requirements engineering. • The technique is extremely helpful developing new systems for entirely new applications.	• In many cases prototypes are expensive to produce in terms of time and cost. • A great problem for prototyping is that the user often resists making changes if once they get experienced.

Table 2. Requirement Prioritisation Techniques

Method	Description	Advantages	Disadvantages
Value-oriented prioritisation method	Prioritises requirements based on their contribution to the core business values and their perceived risks. The first step in setting up a value-oriented prioritization process is to establish the framework and this framework is used to identify the value of the business and the relative relationship of those values. Business values are established at the level of organization. After indentifying the core values, the organization must provide some indication of importance of those values to the organization. This is accomplished by assigning weights that use a simple ordinal scale ranging from 0(not important) to 10(critical).		As prioritisations involve a small subset of stakeholders; the results are biased towards the perspective of those involved in the process.
Pairwise comparison approach	Requirements engineers compare two requirements to determine the more important one, which is then entered in the corresponding cell in the matrix . The comparison is repeated for all requirements pairs such that the top half of the matrix is filled. If both requirements are equally important, then they both appear in the cell. Then, each requirement is ranked by the number of cells in the matrix that contain the requirement.	Pairwise comparison is simple.	Since all unique pairs of requirements need to be compared, the effort is substantial when there are many requirements. Prioritising n requirements needs $n\times(n-1)/2$ comparisons . Hence, a project with 100 requirements would require 4,950 comparisons.
Analytic	The analytic hierarchy process (AHP)	On the other hand	On the one hand AHP is a

Hierarchy Process (AHP)	is a decision-making method. Using AHP to prioritize software requirements involves comparing all unique pairs of requirements to determine which of the two is of higher priority, and to what extent.	AHP is very trustworthy since the huge amount of redundancy in the pairwise comparisons makes the process fairly insensitive to judgmental errors. Another advantage is that the resulting priorities are relative and based on a ratio scale, which allows for useful assessments of requirements.	demanding method due to the dramatically increasing number of required pair-wise comparisons when the number of requirements grows.
100-point test	Each stakeholder is given 100 points that they can distribute as they desire among the requirements. Requirements that are more important to a stakeholder are given more points. Requirements are then prioritised based on the total points allocated to them.	100-point test incorporates the concept of constraint in the stakeholder's prioritisation by giving each of them a limited number of points.	It can be easily manipulated by stakeholders seeking to accomplish their own objectives . For example, stakeholders may distribute their points based on how they think others will do it. In addition, it is difficult for stakeholders to keep an overview of a large number of requirements.
Hierarchical cumulative voting (HCV)	Enables prioritisations to be performed at different levels of a hierarchy. Stakeholders perform prioritisation using 100- point test within each prioritisation block. The intermediate priorities for the requirements are calculated based on the characteristics of the requirements hierarchy. Final priorities are calculated for all requirements at the level of interest through normalisation. If several stakeholders have prioritised the requirements, their individual results are then weighted and combined.	The hierarchical prioritisation in HCV makes it easier for the stakeholders to keep an overview of all the requirements	The prioritisations need to be interpreted in a rational way as stakeholders can easily play around with the numbers.
Requirements triage method	In the requirements triage method, Davis proposed that stakeholders should be gathered in one location and group voting mechanisms used to prioritise requirements. One method to collect group vote is to use the show of fingers to indicate the stakeholders' enthusiasm for a requirement.		A disadvantage is the relative priorities of requirements depend on the stakeholders who attended the prioritisation meeting, and dominant participants may influence the prioritisation.
Win-win approach	In the win-win approach proposed by Boehm, stakeholders negotiate to resolve disagreements about candidate requirements. Using this approach,	Win-win negotiations encourage stakeholders to focus on their interest	The approach is labour intensive, particularly in large projects.

	each stakeholder ranks the requirements privately before negotiations start. They also consider the requirements they are willing to give up on. Stakeholders then work collaboratively to forge an agreement through identifying conflicts and negotiating a solution.	rather than positions, negotiate towards achieving mutual gain, and use objective criteria to prioritise requirements.	
Binary search tree (BST)	In BST, a requirement from the set of requirements is selected as the root node. Then, a binary tree is constructed by inserting less important requirements to the left and more important ones to the right of the tree. A prioritised list of requirements is generated by traversing the BST in order. The output is a prioritised list of requirements with the most important requirements at the start of the list, and the least important ones at the end.	This method is simple to implement	Provides only a simple ranking of requirements as no priority values are assigned to the requirements.

6. REFERENCES

[1] Joachim Karlson, Claes Wholin, Bjorn Regnell "An evaluation of methods for prioritizing software requirements".

[2] Aaqib Iqbal, Farhan M, Khan, Shahbaz. A. Khan "A Critical Analysis of Techniques for Requirement Prioritization and Open Research Issues". International Journal of Reviews in Computing 2009 IJRIC

[3] Goguen, J. and Linde, C. (1993), "Techniques for requirements elicitation", Requirements Engineering, IEEE, 152-164.

[4] Kotonya, G. and Sommerville, I. (1998), "Requirements Engineering: Processes and Techniques", John Wiley, 1998.

[5] "Knowledge Elicitation Techniques: Comparison of Three Methods", Proceedings of the Second International Conference on Requirements Engineering (ICRE96), Colorado Springs. CO, April 15-18, 1996. pp. 4-11

[6] Davis, C. J., Fuller, R. M., Tremblay, M. C., & Berndt, D. J. (2006). "Communication challenges in requirements elicitation and the use of the repertory grid technique". Journal of Computer Information Systems, 78.

[7] Pitts, M. G., & Browne, G. J. (2007). "Improving requirements elicitation: An empirical investigation of procedural prompts". Information Systems Journal, 17, 89-110.

[8] Shaw, M. (2001) "The coming of age of software architecture research" Proceedings of the 23rd International Conference on Software Engineering (ICSE'01), pp: 657-664