Software Engineering, ISSN: 2229–4007 & ISSN: 2229–4015, Volume 1, Issue, 1, 2010, PP-01-05

Social-organizational participation difficulties in requirement engineering process: A study

Dhirendra Pandey[1], Suman U.[2], Ramani A.K.[2]
[1]Department of Information Technology, Babasaheb Bhimrao Ambedkar University, Lucknow, 226025, India
prof.dhiren@gmail.com
[2]School for Computer Science & IT, Devi Ahilya Vishwavidyalaya, Indore, MP, India,
ugrasen123@yahoo.com, ramani.scs@dauniv.ac.in

Abstract- Requirement engineering is a technique for analyzing and documenting the requirement of users. Requirement engineering is a necessary prerequisite to build a healthy participation between customers and the organization for better understanding the requirements from customers by the software developers. The requirements engineering phase of software development process is characterized by the strength and significance of participation activities. During this phase, the various stakeholders must be able to communicate their requirements to the analysts and the analysts need to be able to communicate the specifications they generate back to the stakeholders for validation. This paper describes a field investigation into the problems of participation between disparate communities involved in the requirements specification activities. The results of this study are discussed in terms of their relation to three major participation barriers: 1) ineffectiveness of the current participation channels; 2) restrictions on expressiveness imposed by notations; and 3) social and organizational barriers. The results confirm that organizational and social issues have great influence on the effectiveness of participation.
Keywords: Requirement engineering, software project, participation difficulties, questionnaire, interview.

1. Introduction

Requirement specification is an important dimension of requirement engineering process which is based on domain understanding, i.e. organizational, technical, functional, and social. Ideally, the requirements team members are selectively recruited so that both the levels and distribution of knowledge within the team cover all aspects of the domain [1]. However, this is seldom the case because of knowledge shortfalls such as the thin spread of application domain knowledge in most organizations [2]. In general, individual members do not have all the knowledge required for the project and must acquire additional information before accomplishing productive work [3]. Knowledge acquisition and sharing can only be achieved through effective participation between the various stakeholders. It is widely recognised that participation problems are a major factor in the delay and failure of software projects [2].

This is especially true of "socio-technical" software systems, which must exist in a complex organizational setting. The organizational domains into which such software is introduced are often too intricate and fluid to be fully understood. In this situation, misunderstandings and conflicting views are rife. There have been a number of field studies into software engineering in general and requirements engineering in particular [4]. Our study differs from previous investigations in that it focuses on the participation characteristics of the requirements engineering process. Moreover, our investigation not only utilized the experience of software engineering practitioner, it also reflects the views and experiences of end-users based on their recent software procurement projects. The domain of our study was the requirements engineering phase of fully customized software systems development projects. The field study was conducted in two stages using two data gathering methods like interviews and questionnaires.

2. Research Method

A combination of learning, data gathering and analysis techniques were applied to investigate the participation problems, their causes and consequences. The two principle sources of information were the literature and the empirical study. The ever growing literature on software engineering in general and requirements engineering in particular was surveyed to gather information about the software development problems, especially those that occur in the early phases, and the sort of tools and techniques that were or are being developed to overcome these problems. A cross section of social science and computer supported co-operative work (CSCW) literature was also surveyed to help in the analysis of the empirical results and reasoning about the possible causes and consequences of participation difficulties.

2.1 Empirical Work

The aim of the empirical part of this research is to provide material for hypotheses, to aid the identification and reasoning about the participation difficulties and their causes and consequences. Although there are inherent complexities in combining qualitative and quantitative methods, it was decided that such an empirical base was essential to avoid

unsupported assertions. The empirical work was carried out in two stages. The first consisted of informal interviews and observations to establish some knowledge about practices and methodologies of both developers and their customers. These interviews concentrated mainly on the participation channels between agents participating in any software development project, as well as on the problems that can be attributed to the ineffectiveness of those participation channels. Other management and technical issues were also discussed. Most of these interviews were taped for further analysis and reference.

The second stage of our empirical work was based on two questionnaires; one for customer and one for the software developers. These questionnaires were designed to get a quantitative evaluation for the various aspects of the participation activities during Requirement Engineering. The developers were all involved in either developing a new software system or maintaining an existing one. Some were also involved in the provision of hardware systems. Questionnaires were sent to some companies in the India. Responses represent a cross-section of the companies that were targeted.

The interviews showed that practitioners, no matter how experienced, found it easier to be precise about facts and procedures than about opinions and judgments. Therefore we could not simply ask for direct judgments for each of our hypothesis. Instead, each hypothesis was tested in terms of its outward effects and indicators. Most of these turned out to be multi-dimensional and thus had to be measured through more than one indicator. For example, the effect of organizational power was multi-dimensional in that it governs both the choice participants and their working procedures. It had to be tested in two separate questions each of which had a number of variable answers. In order to get a value for the strength of feeling for each indicator, we employed a Likert Scale method, also known as semantic differential [4]. For each variable we used one to five values of strength in relation to the other variables within the same question.

3. Participation Difficulties

Large software projects suffer serious breakdowns in co-ordination and participation throughout their development life cycle. In this section we present some of the causes for the breakdown of participation during the requirements engineering phase of software development projects.

3.1 One Way Participation Channels

In many ways, software engineering methodologies are participation methodologies. Much emphasis is placed on the notations used to convey information both within the development team and with the various stakeholders. Ideally, the channels of participation between these various communities would be perfect, so that all knowledge is shared. In practice, it is expensive and time-consuming to support extensive participation between the communities, and the channels are restricted to one way participation in the form of specification documents. Some researchers have observed that documentation is ineffective for participation, as it does not help resolve Misunderstandings [2]. Nevertheless, an implicit “over-the-wall” model exists in most software development projects: at each stage in the project, a specification is thrown over a wall to the next team who are waiting to proceed with the next phase. The metaphorical wall is sometimes encouraged by management practices, but more often is merely a result of the practicalities of coordinating a large team. The results of this study showed that specification documents are still the most common format in which analysts communicate requirements back to their clients for validation (see table-2).

Q- In what format did you get the analyst’s interpretation of your requirement?

There are two standard approaches to this problem. The first emphasizes the development of better notations, and effective use of electronic repositories. The second emphasizes the importance of contact between the development team and other stakeholders, and has given rise to practices such as end-user participation and ethnographic techniques. Each of these approaches has its own set of problems, and neither directly addresses the question of facilitating appropriate and effective participation over restricted channels. Our study showed that practitioners find it easier to adapt a compromise of the two approaches by enriching notations with natural language descriptions and by utilizing the personal contact of face-to-face discussions.

3.2 Familiar Participation

Organizations are, traditionally, described in terms of an *Organizational chart*. This is often the first thing handed out to anyone inquiring about the structure of the organization. However, many important power and participation relationships are not represented in the organizational chart. One of the researchers [6] makes an analogy between the organizational chart and a road map, where the map is invaluable for finding towns and their connecting roads, but it tells us nothing about the economic or social relationships between the regions. Although, very useful in terms of providing information about formal authority and the division of organizational units, the organizational chart does not tell us

anything about the informal relationships that exist in every organization.
During the interviews that we conducted with practitioners, they outlined many difficulties that are caused by unexpected interactions between elements of the system, be it software modules or humans. In spite of the time and effort spent on studying organizational structures and the flow of power and information through them, our subjects admit that they can never account for all possible interactions and often have to backtrack as a result of discovering a new relation or line of participation that has to be incorporated into the system. On the other hand, such informal participation channels can be destroyed by rivalries and animosities, which can discourage co-operation and affect the normal flow of information.

Q- How do you exchange information with the software development team?

3.3 The lost link

The inability to trace the human sources of actual requirements and their related information is identified as the crux of the requirements traceability problem [7]. Requirements Traceability is vital for all phases of the software development cycle to aid reasoning about requirements and justify changes. Our study showed that the traceability problem is particularly serious in the later stages of requirements engineering and in cases when new requirements are introduced late in the project life cycle. In order to check that newly introduced requirements do not conflict with existing requirements, it is often necessary to re-establish participation with the human sources of existing requirements. This is particularly problematic, because by this time the analysts may have reduced, or even halted, contact with the end-users of this software and may even have started working on a new project, while their programmers get on with later phases of this project.

Q- How do you established traceability and responsibility for requirements?

Table 4 presents the links that practitioners use in order to establish requirements traceability to the requirements sources. We can see from the above results that most analysts link the requirements only very generally to user groups and departments, which means that traceability is not direct. Only 15% linked the requirements to their individual sources by name, which means that those sources can be traced directly if necessary. The link to job titles might sound like a good idea, but it does not work in dynamic organizations where people move between jobs and even move between organizations.

3.4 The Notations conflict

While programmers, software engineers and analysts are happy talking about the system in terms of it procedures and data structures, end-users prefer to talk about the system in terms of its general behavior, functionality and applications. The different communities involved in the specification process prefer different types of notation, and various people will be unfamiliar with various notations. For example, a user will not want to learn to read formal specification languages, but the programmer may require these to obtain an appropriate level of detail. It is often the analyst's responsibility to choose the notations that will best describe the system for each interest group. Thereafter, the chosen notations are used to explain the system differently to each group. In doing so, the analyst combines the notations with other explanation techniques, to make notations easier to read and understand.
The choice of the explanatory tools utilized by the analysts and the extent to which they are needed depend on the notation used and the audience's familiarity with the notation. Typically, two types of knowledge are used as a high level framework to anchor detailed knowledge: the control flow information, which might be represented by specialized notations such as pseudo code and flowcharts, and data structure information, which might be represented using diagrams or a textual description [8]. Some software programs such as the traditional numerical analysis systems have complex control structure with relatively simple data structures. On the other hand, traditional commercial applications have complex data structures with relatively simple control flow. Some of the researchers [9] conducted an experiment to compare comprehension with nine forms of program description including natural language, a program design language, flowcharts and hierarchical diagrams. They found different results for different types of questions, but no particular style appeared to dominate. However, in their study of program coding from the nine notations, two researchers [10] found that the program design language and the flowcharts diagrams were more helpful than natural language descriptions.

Regardless of the chosen notations, most users express their requirements in natural language. Then it is the job of the analyst to translate requirements statements into some kind of representational objects in a domain model. Once the requirements are modeled, they are presented to end-users for validation. At this stage the analysts are faced with another participation problem when end-users are not

Software Engineering, ISSN: 2229–4007 & ISSN: 2229–4015, Volume 1, Issue, 1, 2010

familiar with the notations used to model their requirements. On the other hand, when analysts, under pressure to keep up with the project schedule, pass raw natural language requirements to programmers, then time is wasted in trying to interpret them. In one case he had to read over a page of text to understand the requirements for a screen layout for a particular database form.

Qu- How often do you use each of the following methods to help client understand the representation of their requirements?

5. Conclusion

There are lots of difficulties in trying to make effective participation channels between social and organizational elements. One of the dangers is that each community interprets things in the light of their own background assumptions. This is especially problematic with non-interactive participation, such as specification documents, where there is no opportunity to check that the reader has interpreted them as was intended. Some of the researchers [11]. Points out a fundamental problem to do with the participation of abstract concepts, in that requirements specifications "document what it is that the analyst thought it was that the problem owner said he thought he might want". The uncertainties that McDermid describes propagate and multiply at each exchange of information. Some researchers uses the term "Ontological Drift" to describe the change in meaning of abstract terms as they are passed between different communities[12]. In this paper we noted that documents are a poor substitute for interpersonal participation. A pressing and practical problem is to find-out more about the communicational weaknesses of current notations and methods so we can accommodate for their weaknesses. For each concern, we need to determine what types of question that concern may wish to make of a description produced in a notation. This can only be achieved by observing the meetings and conversations in which descriptions are referred to. The practical implications of theses findings include: indicating where and how organizational power is used, outlining the extent to which software practitioners rely on documents as the main participation medium, revealing the dangers of the technical gap between the two main communities and presenting informal participations as the means for bridging that differences.

References

[1] Amer Al-Rawas1 and Steve Easterbrook (1996*) Proceedings of the First Westminster Conference on Professional Awareness in Software Engineering, Royal Society, London*.

[2] Curtis B., Krasner H. & Iscoe N. (1988) *Participations of the ACM,* 31(11), 1268-1287.

[3] Walz D., Elam J. and Curtis B. (1993) *Participations of the ACM* 36(10), pp.63-77.

[4] Curtis B. (1990) *In Diaper et. al. (Eds), Human-Computer Interaction - INTERACT '90, Elsevier Science Publishers, North-Holland*. 35-40.

[5] Frankfort-Nachmiais C. and Nachmiais D. (1992), *4th Ed., Edward Arnold*.

[6] Mintzberg H. (1979) *Prentice-Hall.*

[7] Gotel O. and Finkelstein A. (1994) *Proceedings of the First IEEE International Conference on Requirements Engineering, Colorado springs,*.94-101.

[8] Shneiderman B. (1982) *In Ledgard, H. (Eds), Technical Notes: Human Aspects of Computing,Participations of the ACM* 25 (1), 55-63.

[9] Sheppard S. B., Kruesi E. and Curtis B. (1981) *Proc. 5th Int. Conference on Software Engineering, San Diego, CA, available from IEEE*,1981, 207-214.

[10] Sheppard S. B. and Kruesi E. (1981) *Proc. Trends and Applications: Advances in Software Technology. Held at NBS, Gaithersburg, MD, available from IEEE*, 7-13.

[11] McDermid J. A. (1993) *In M. Bickerton & M. Jirotka (Eds.), Requirements Engineering. London*: *Academic Press*.

[12] Robinson M. and Bannon L. (1991*) In L. Bannon, M. Robinson, & K. Schmidt(Eds.), Proceedings of the Second European Conference on Computer- Supported Co-operative Work (ECSCW-91), 25- 27 September, Amsterdam, the Netherlands*. 219-233.

Table1- Interview sample size and questionnaires responses

Study	Developers (S/W Practitioners)	Customers (End-users of the software)
Interview Sample Size Questionnaires Responses	5 42	5 37 (21 Participating and 16 Non-participating)

Table2 -The formats in which requirements are communicated

Formal specification documents	40%
Informal combination of documents and discussion	30%
Informal specification documents	11%
Natural language discussion	19%

Table 3- Methods in which end-users exchange information with analysts

Face to face discussion	45%
Formal meeting	25%
Telephone conversation	12%
Using documents	18%

Table 4- Links that software practitioners use to trace requirements sources

Requirements are linked to user group and departments	65%
Requirements are linked to individual source by name	15%
Requirements are linked to job titles of their source	20%

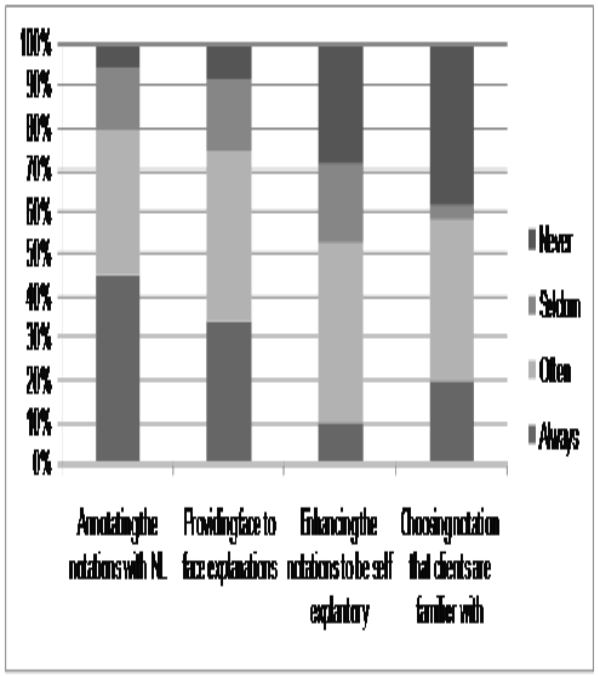

Fig. 1-Methods of providing additional explanation.