Conceptualizing Software Engineering “People” Issues

Medha Umarji, Carolyn Seaman

Dept. of Information Systems
Univ. of Maryland Baltimore County
Baltimore, USA
{medha1, cseaman} @umbc.edu

Abstract. The Guide to the Software Engineering Body of Knowledge (SWEBOK) comprehensively covers most of the Software Engineering literature. In this paper we propose the addition of a knowledge area pertinent to the organizational, individual and methodology-related aspects of Software Engineering. Such a conceptualization is required because these issues are inherently important and relevant to existing knowledge areas and for the overall success of a software project. The issues that are outlined are by no means exhaustive and the primary intention is to provide a starting point for reflection on these and similar factors. We then provide an example of a knowledge area and how these concepts are applicable to it.

1 Introduction

The knowledge areas of the Guide to the SWEBOK mostly correspond to topics in software engineering that have received significant attention in the research literature and that have achieved a certain degree of ‘maturity’ – in that they have established norms and practices and a good foundation for further research. The Guide to the SWEBOK is a comprehensive and detailed compilation and its use is facilitated by the taxonomies that are created to represent each knowledge area.

There are, however, some emerging research areas that have been mentioned intermittently in the Guide but have not yet been addressed at length. Organizational and personnel-related aspects of software engineering fall into this category. It is important to know more about these issues because an increasing amount of empirical evidence points us in this direction [2], [3], [4], [5], [7], [8]. Also, we believe that there is a need to change what is made available to the practitioners today in terms of software engineering knowledge, and make them realize the importance of these issues and incorporate them into everyday project development and management practices.

However, these “people” issues or “soft” issues as they can be called, pervade all areas of software engineering, impacting all projects and practices. Therefore, it is difficult to group them under one category and still establish their relevance to each aspect of software engineering. On the other hand, they do have commonalities and specifying their inter-relationships under each topic may cause redundancy.

Therefore, we attempt to make a conceptualization of the *individual, organizational* and *methodological* issues that are pertinent to software engineering personnel and practices. We propose to list the main factors, their inter-relationships and causality amongst these factors. The conceptualization is generic so that specific instances can be derived and tailored to the software engineering area under consideration.

We intend to characterize this conceptual framework with domain-specific knowledge, experience and empirical evidence.

One interpretation of the term "ontology" is that it is a depiction of generic relationships between entities in a domain. It should be possible to represent all the entities and their inter-relationships with the ontology specification. We believe that it would be premature to call our proposed conceptualization an 'ontology', but it is an effort in this direction. We then follow it up by an example of a knowledge area and how these concepts are applicable to it.

2 Classification of Issues

We present below a classification of issues based on organizational, individual and methodological aspects of software engineering. This list is by no means exhaustive, and the primary intention is to provide a starting point for reflection on these and similar factors.

2.1 Organizational Issues

For the success of a project, there are a number of different factors that come into play and organization-dependent variables are very important in this context. By organizational issues, we mean characteristics of the organization and its culture that may be influenced by the nature of the industry, corporate goals, location and resource availability.

Visibility of a project: Software developers tend to get very involved in their project or their part of the project and lose the holistic view of their project within the organization. The organization is perceived as an external entity whose welfare is not relevant to the project at hand. Due to this, the developers may have trouble understanding the goals and direction of the project and this may hamper project development. Hence, "visibility" of a project within the right context is essential and is a very important organizational variable.

Organizational and national culture: The culture of an organization is influenced to a large extent by the culture of the country in which it is based. Especially in the software industry this is an issue because many organizations outsource their software development to foreign countries and might have difficulties in maintaining the same standards and work practices as they do locally. The extent of cultural influence needs to be investigated more to better understand the functioning of the organization and its personnel.

Communication channels: Communicating regularly the status and direction of the organization to the employees is the responsibility of the management. Lack of good communication channels can severely affect the work culture in the organization.

Exposure to a wide variety of new technologies, tools and systems: The absorptive capacity of an organization increases as its personnel are exposed to a wide range of new techniques and systems. The "openness" of the management to failures and successes also helps developers to learn more about applying the tools and technologies to their work and using them more effectively in their project.

Empowerment: Giving developers flexibility to take initiative can help to increase their contribution to a project so it should be a part of organizational policy.

Transparency of process: Transparency facilitates understanding and traceability within the organization and increases the developers' feeling of control.

Availability of a help facility: Having reliable, experienced personnel, good library resources and access to domain experts is also a responsibility of the organization.

Nature of the business: The nature of the software business, e.g. governmental, commercial, in-house development, etc., has an effect on the organizational culture because this determines the nature of the projects. For example, software development for a mission-critical defense-related project may have to be defect-free, but schedule may not be a pressing concern.

Rate of staff turnover: The software industry has progressed very rapidly and technically skilled professionals are constantly in demand. This is one of the reasons that software professionals switch jobs often. The problem with this is that the organization loses valuable experience and 'lessons learned' in addition to good people. Therefore it is important for an organization to retain its employees.

2.2 Individual Issues

By individual issues, we refer to the immediate context of a person and its effect on his/ her perceptions and behavior in the workplace. We also discuss some of the general desirable characteristics required for software personnel.

Code ownership: The feeling of code ownership also plays a role in many of the phases of software development such as testing and reviews. It is definitely important to be considered especially while designing processes and allocating work and explaining work to people.

Communication skills: Developers are highly technically skilled but they may lack good communication skills. The nature of their job is such that they need to interact in teams very often and carry out lengthy discussions with their colleagues and superiors. Some people are unable to articulate their thoughts precisely and clearly and lack of these skills becomes the root of problems in the workplace.

Fear of adverse consequences: This may be triggered by changes in the workplace that are perceived as threatening to an employee's position in the company and to the resources allocated to him/her. This can affect the employee's motivation to perform and management should therefore safeguard against this.
Attitude: Attitude is the assessment of the desirability of performing a behavior, by an individual. Attitude has two sub-components: affective (e.g., happy-sad) and cognitive (e.g., beneficial-harmful). Attitude is also influenced by the job seniority, experience and management policies [1], [6], [9].

Perceived Behavioral control (PBC): This is borrowed from the Theory of Planned Behavior (TPB) [1], [6] and refers to both personal and situational impediments to performing a behavior. It addresses the situation where performance may depend on non-motivational factors such as availability of requisite opportunities and resources. PBC includes both external and internal factors. External factors are those things outside one's person that may affect behavior; therefore, we include here the user resources such as adequate time and financial stability. Internal factors refer to the personal ease or difficulty of performing the behavior, based on one's prior experiences, skills gained through learning, and intelligence. PBC also has an impact on the overall feeling of control and motivation.

Subjective Norm: Subjective Norm is also defined in the TPB, [1] and has to do with the attitudes of a group of co-workers. It plays a significant role in any workplace. If one developer has negative opinions about the organization, the management or the work, they may influence people around them and this attitude begins to spread through the group. This would potentially cause a slack in the general energy and interest. The descriptive norm is one component of Subjective Norm. It is the notion that if people within the social network of a subject perform a certain behavior, then that subject's motivation to perform that behavior is positively enhanced. Injunctive norm is the other component of Subjective Norm, and it is the notion that if people within the social network of a person *want* him/her to perform a certain behavior that they consider to be useful, eventually his/her motivation for performing that behavior will be positively influenced. Also, people tend to believe their managers, their superiors, and experienced colleagues more than theory or books.

2.3 Methodology-related issues

In this category we list variables that are characteristics of methodologies and vary with different methodologies. In the context of the Guide to the SWEBOK, we basically refer to the knowledge areas such as testing and maintenance, as "methodologies", as well as new approaches to these areas that might be introduced into an organization, such as mutation testing, or software measurement.

Nature of the methodology: Nature of the methodology can be determined based on the degree of change induced, intrusiveness into work practices, amount of training required and such other factors.

Learning curve: Learning curve of the software engineering methodologies such as metrics programs or CASE tools should be taken into account while charting schedules and allocating work to people. It takes time, especially to learn to build these techniques into the regular schedule and work practices.

Usefulness: The usefulness of a methodology is an important consideration [6], [9]. Unless a methodology adds value to the organization and work practices of the people it will not be used effectively and well.

Ease-of-Use: The degree of complexity or difficulty of a methodology also impacts the motivation to use it and the performance of people using it [6], [9].

3 Example (Software Maintenance)

We envision that the sets of factors listed above can be incorporated into each knowledge area of the Guide to SWEBOK. To illustrate this, we discuss below an example of Software Maintenance and how some of the issues discussed so far can be applied to this domain.

3.1 Organizational Issues

Maintenance of a software system is facilitated by the availability of relevant, concise and up-to-date documentation. However, producing this kind of documentation is not always an easy task, and in the pressure of meeting deadlines, and tight schedules, this process may get neglected. This is where the organization policy comes into play.

An organization's management has to communicate the policies regarding good documentation, at the start of the project. Therefore *communication channels* are very important. The organization can provide a developer the flexibility to work on this, by allowing more lee way in the schedules.

Also, the *availability of a help facility* like an experienced developer who has worked on a system, greatly aids the work of software maintainers, while working on that system.

The *rate of staff turnover* affects the availability of a help facility. If turnover is high, the number of experienced personnel available to maintainers reduces, making it difficult for them to get up-to-date information they need about a system.

3.2 Individual Issues

The feeling of *“code ownership”* is also important as software maintainers usually do not have a feeling of “code ownership” and sometimes this hampers the in-depth understanding and level of comfort while working with the system.

Software maintainers need to have good *communication skills* because they may have to often explain a problem to people that were involved in the development of the original system in order to gain insights into making modifications to the system. Maintainers have to be able to express themselves well and be able to elicit the information they need from the original developers.

3.2 Methodology-related Issues

The *nature of maintenance work* also has an effect on the work practices of software maintainers. Maintainers usually work with code written by other people, therefore the difficulty of their task depends upon factors like how much documentation is available, presence of helpful comments in the code, and the type and volume of changes required to be made.

The *learning curve* for maintenance tasks largely depends on how much time it takes to get familiar with a system. Familiarity implies that a change can be made to the system without having any adverse effects on some other parts of it.

4 Conclusions

From the above discussion, we can deduce that each phase of the SDLC, and possibly the other SWEBOK knowledge areas, can be analyzed from different aspects with the help of the factors outlined above. Also, other than just the technical aspect of software maintenance, the organizational context and individual issues could determine whether it is implemented successfully or not. While creating this representation of software maintenance, we had to add some factors to our initial list of factors. This list may get longer as we work with the other phases of the SDLC or other knowledge domains in software engineering. However, many factors will keep repeating and at some point, we should have close to an exhaustive list of factors. Therefore, we claim that a similar and more detailed representation should be created for the other areas and this should be made available to practitioners so that they can enhance their understanding of these issues and act upon them.

References

1. Ajzen, I.: The Theory of Planned Behavior. Organizational Behavior and Human Decision Processes, Vol. 50, (1991) 179-211.
2. Brooks Jr., F.: The Mythical Man-Month. Reading, Mass.: Addison-Wesley (1975).
3. Brown, N.: Industrial-Strength Management Strategies. IEEE Software, Vol. 13, no. 4, (1996) 94 - 103.
4. McConnell, S.: Rapid Development: Taming Wild Software Schedules. Redmond, Washington: Microsoft Press (1996) 249-272.
5. Curtis, W.: By the Way, Did Anyone Study Any Real Programmers? Empirical Studies of Programmers. Norwood, N.J.: Ablex Publishing (1986) 256-262.
6. Davis, F.: Perceived Usefulness, Perceived Ease of Use, and End User Acceptance of Information Technology. MIS Quarterly, Vol. 13 (1989) 318-339.
7. Seaman, C.: Qualitative Methods in Empirical Studies of Software Engineering. IEEE Transactions on Software Engineering. Vol. 25, no. 4 (1999) 557-572.
8. Seaman, C., Basili, V: Communication and Organization: An Empirical Study of Discussion in Inspection Meetings. IEEE Trans. Software Engg. Vol. 24, no. 7, (1998) 559-572.
9. Venkatesh, V., Davis, F.: A Theoretical Extension of the Technology Acceptance Model: Four Longitudinal Field Studies. Manage. Sci., Vol. 46, no. 2, (2000) 186–204.