A Proposal for Enhancing User-Developer Communication in Large IT Projects

Ulrike Abelein, Barbara Paech
Institute of Computer Science, University of Heidelberg, Im Neuenheimer Feld 326, 69120 Heidelberg, Germany
{abelein, paech}@informatik.uni-heidelberg.de

***Abstract*—A review of the literature showed that the probability of system success, i.e. user acceptance, system quality and system usage, can be increased by user-developer communication. So far most research on user participation focuses either on early or on late development phases. Especially large IT projects require increased participation, due to their high complexity. We believe that the step in software development when user requirements are translated (and thus interpreted) by developers into a technical specification (i.e. system requirements, architecture and models) is a critical one for user participation. In this step a lot of implicit decisions are taken, some of which should be communicated to the end users. Therefore, we want to create a method that enhances communication between users and developers during that step. We identified trigger points (i.e. changes on initial user requirements), and the granularity level on which to communicate with the end users. Also, representations of changes and adequate means of communication are discussed.**

***Keywords*—user-developer communication; task-oriented requirements engineering; rich media theory**

I. INTRODUCTION

Large IT projects based on COTS software are becoming more common than individual developments, as companies move away from bespoke development and rather purchase packaged software (SW). Projects introducing those SW systems are rather complex due to the required customization for company-specific processes. Also, most companies are still using traditional project management and SW development methods like the waterfall model ([1] cited by [2]). Their advantages are high stability and clear agreements on price, timeline and scope [3]. However, the drawbacks are long periods of waiting for the business side due to long development cycles [3]. Within these cycles requirements transform, as the translation from user to system requirements leads to a lot of interpretation and misunderstanding in combination with a low level of user-developer communication (UDC). There are two effects of these long cycles: First, end users do not feel integrated in the project. Second, end users do not recognize their requirements in the acceptance phase, due to a high level of transformation and a long time span between elicitation and validation [4]. Both effects lead to a low acceptance of the SW and a low motivation to participate in large IT projects. A review of literature showed lots of evidence that UDC has positive effects and can lead to higher user acceptance, system quality and usage, thus to higher system success [5–7]. The measures for system success are adopted from [8], showing that these are mostly used as indicators for system. success. The topic of user participation (UP) has been widely researched especially in the area of information systems, but it is still an open question of how user involvement should be integrated into system development [9]. Other known methods found in literature for UP often do not clearly define how exactly (i.e. in which phases, with which content) participation should take place [10]. Also, most methods focus on how to get information for requirements from users, but not on how and when to communicate changes in requirements (or in their technical realization) if they are transformed during development. One can argue that agile approaches implicitly use that sort of communication, as they claim very close cooperation [11]. However, they do not work well in large IT projects, as the end user is not constantly on site [2]. As there is a positive correlation between UDC and system success [5–7], as well as a lack of methods that focus on that, we believe that a new method to enhance UDC is required when translating user requirements ('a statement […] of what services the system is expected to provide and the constraints under which it must operate' [12]) into technical specifications. We allocate (according to the definition in [12]) system requirements, system architecture and system models into the technical specification.

In Section II, we describe the motivation, background and first ideas for our method. We conclude and state our open questions for further research in Section III.

II. APPROACH TO ENHANCE UDC

In this section we first motivate our approach, and then we sketch Task-oriented Requirement Engineering (TORE) [13] on which our method is based. Afterwards, the four aspects *trigger points to start communication, granularity level for communication, representations of changes* and *communication means* are described.

A. Motivation for the Approach

Most existing research on UP focuses either on the early development phases, e.g. elicitation of user needs, or on the end of the project, e.g. on the user acceptance test [10]. We believe that in large IT projects using traditional development methods, there is a need for enhanced UDC focusing on the translation process from user to system requirements. An interesting study on the effects of

communication gaps in large IT projects [14] supports that. [14] found out that such gaps are caused, beside others, by complex products, large organization and an unclear decision structure. They identified different effects of missing communication: unmet customer expectations, low motivation to contribute to the requirements work, and developers controlling what is implemented. Also, the requirement coverage is unclear due to the lack of discussions between the design and requirement teams regarding changes. Especially the last effect supports our view that the transfer of user requirements into more technical specification requires attention. At this point a lot of interpretation is involved. For example a requirement specifies the results of a system, e.g. the invoice must be delivered to the customer via email, but not how exactly this is implemented. Thus, the translator does take a lot of implicit decisions in this step, some of which should be communicated to the end users (see suggestion for relevant changes in Table 1). In our example the decision "Will the email always be sent as the last step of a workflow based system or is it possible to send an email after the invoice generation" is quite relevant for the user. Therefore, we want to create a method with a special focus on the translation from user requirements into technical specification.

B. Background: Task Oriented Requirements Engineering

Our approach of explaining the required decisions for the translation of user requirements into a technical specification is based on the TORE method [13]. TORE has been developed by one of the authors and others. By this method, 16 different implicit or explicit decisions on the behavior of the system are defined (i.e. we do not yet consider non-functional requirements as they are orthogonal to the proposed abstraction levels of TORE, but we plan to extend our method to non-functional requirements in our further research). The decisions are grouped in four abstraction levels: *Task level* – decisions about the roles and tasks to be supported by the system. *Domain level* – decisions on the activities to be supported by the system and the domain data relevant for these activities. *Interaction level* – decisions about the distribution of activities between humans and computers aligned with decisions on user interface (UI) structure. *System level* – decisions about the internals of the application core and the graphical user interface (GUI). The first level requires only one decision about user roles and their task. The domain level comprises four decisions: determination of the relevant as-is activities, definition of to-be activities, system responsibilities (here we will use the more prominent term feature) and decisions on the relevant domain data. The interaction level comprises also four decisions: system functions, user-system interaction, interaction data for input and output of the system and structure of the UI. Lastly, the system level has two different decision clusters on the core application (high-level application architecture, internal system actions, and internal system data) and on the GUI (navigation and support functions, dialog interaction, detailed UI-data, and screen structure). We will use the outlined decisions to structure our method.

C. Ideas for a Method to Enhance UDC

First, it needs to be identified, which content is important for the end user: which are reasonable points to start communication? Those trigger points can be decisions taken in the translation or changes on agreed user requirements. As shown in Table 1 the trigger points correspond to a subset of the TORE decisions and thus can be aligned to the TORE levels. As we focus not only on SW development but also on project management, we extend them by the project level (including decisions regarding cost, schedule and scope of the project). Also, we introduce the business process level, which comprises decisions about functionality or features composed in business processes. The trigger points will vary with different roles and occasions. Therefore, we suggest to use a RACI (R–Responsible, A–Approved, C–Consulted, I–Informed) matrix [15]. Regarding the roles, we will focus on end users and their management. Developers take responsibility (R) for all decisions listed in Table 1 (one exception can be cost allocations which is explained below), but this is not mentioned explicitly. We also do not list an I for a role, if this role is consulted (C) or approves (A), as an approval and consultation requires information in advance.

As summarized in Table 1, changes in cost are relevant for the management and, depending on the project structure (e.g. the. budget for system development is directly paid by the business unit), managers might be responsible directly. In all cases they need to be consulted and approve the change. We suggest informing the end user so s/he understands resulting changes, but as they are not directly involved, there is no need to consult them or get their approval.

TABLE 1. RACI MATRIX FOR RESPONSIBILITIES

Abstraction level (based on TORE)	***Changes/decisions in…***	**Mgmt. of users**	**End users**
Project level	Cost allocation	(R),A,C	I
	Timing (go-live dates)	A	C
Business process level	Business processes	A	C
Task level	Responsibility of the users	A	C
Domain level	To-be activities	I	A,C
	Features	I	A,C
	Domain data	I	A,C
Interaction level	Workflow of the system	-	A,C
	User Interface (incl. I/O)	-	A,C
System level	Technology	(A), C	I

Changes in timeline, business processes or user responsibility need to be approved by the management. But most of the issues need input from the end user, Changes on the domain level require a lot of domain knowledge to recognize the consequences, thus they need to be approved by the end user. However, to avoid problems with the management, they should be informed. We think there is no need to consult managers, as they will not be interested in detailed discussions. The same is true on the interaction level, as changes of UI or in workflows are not relevant for them. Thus, they should be approved by the end users. As these should not have other dependencies, there is no need to inform the management. Changes or decisions regarding technology should be discussed with the management. Depending on the governance, they might need approval

from the management. Decisions on the system level can have consequences on other levels, thus we suggest informing the end users. We assume that GUIs are designed together with the users and thus do not need additional communication. For all other changes in technical details, we assume that they are not relevant for the end users or their management. The granularity level for the communication with the user is given through the abstraction levels of TORE. We assume that most discussions will be on the domain level (e.g. changes on features) or on the levels above, such as the task, business process or project levels. However, if it comes to changes in workflows or UI we have to step down to the interaction level, e.g. UI structure. Furthermore, it needs to be figured out how the results of the decisions (content) as well as the changes in them can be represented in the discussion with the end user. We suggest using the existing documentation for content representation and highlighting occurring changes in them. A list of possible representation models for each level is listed in [13]. Finally, means of communication need to be specified. According to the media richness theory (MRT), an activity that requires communication needs to be matched to the medium's ability to convey information [16]. [16] distinguish between uncertain and equivocal communication. Equivocal tasks should be managed by rich communication channels; whereas standard data can be handled by leaner channels. Based on MRT, face-to-face communication is the richest channel. Videoconferencing is a bit leaner, but restricts some visual cues. Phone is not capable of transmitting visual cues, but instant feedback is possible. The lowest richness has email and thus is a good fit for communicating well-understood issues [16]. We think changes with impact on the project that need to be approved by the management are equivocal and thus should be discussed in meetings (if possible face-to-face or through videoconferences). Informing the end users is less equivocal, as they do not need to take part in the decision. Thus, it is sufficient to use a leaner communication, such as email or a central wiki. Changes to be approved by the end user are equivocal, thus media rich face-to-face workshops can be a valid medium. If possible, changes should be clustered for a half-day workshop. If workshops are not possible, this can also be mediated through an online meeting place. However, efficiency drops due to missing visual cues. Captured rationale of decisions should be available to all project members including the end users. But in that case, equivocality is less important, therefore a lean medium like a wiki should be used.

III. Conclusion

In this paper we described the need and first ideas for a method to enhance UDC in large IT projects. It can be derived from literature that enhanced UDC has a positive impact on system success, but we did not find a method that focuses on the communication of changes in requirements (or their realization). We identified trigger points based on the TORE abstraction levels [13]. We assume most discussions are performed on the domain level. In terms of representation, we suggest the reuse of existing documentation. To find the most adequate means of communication, we look into the MTR, which suggest using rich data channels in case of high equivocal content.

We have some open questions to answer in our further research. First, we need to define how the rationale of decisions can be represented. Second, we need to detail representations of changes, in order to not only highlight the change, but also to draw comparisons to previous versions. Third, we need to further research non-functional requirements. In addition to the remaining open questions we will also further specify the method (e.g. which decisions are to be communicated?) and validate our approach in case studies to ensure feasibility, if possible in real life IT projects.

References

[1] R. D. Austin and R. L. Nolan, *How to manage ERP initiatives*. Boston: Division of Research, Harvard Business School, 1998.

[2] G. B. Alleman, "Agile project management methods for ERP : how to apply agile processes to complex COTS projects and live to tell about it," in *Extreme Programming and Agile Methods: XP/Agile Universe*, D. Wells and L. Williams, Eds. Springer Verlag, 2002, pp. 70-88.

[3] M. Fowler and J. Highsmith, "The agile manifesto," *Software Development*, vol. 9, no. August, pp. 28–35, 2001.

[4] W. J. Doll and G. Torkzadeh, "A discrepancy model of end-user computing involvement," *Management Science*, vol. 35, no. 10, pp. 1151–1171, 1989.

[5] J. D. McKeen, T. Guimaraes, and J. C. Wetherbe, "The Relationship between User Participation and User Satisfaction: An Investigation of Four Contingency Factors," *MIS Quarterly*, vol. 18, no. 4, pp. 427 - 451, Dec. 1994.

[6] M. J. Gallivan and M. Keil, "The user-developer communication process: a critical case study," *Information Systems Journal*, vol. 13, no. 1, pp. 37-68, Jan. 2003.

[7] T. McGill and J. Klobas, "User developed application success: sources and effects of involvement," *Behaviour & Information Technology*, vol. 27, no. 5, pp. 407-422, Sep. 2008.

[8] M. I. Hwang and R. G. Thorn, "The effect of user engagement on system success: A meta-analytical integration of research findings," *Information & Management*, vol. 35, no. 4, pp. 229-236, Apr. 1999.

[9] S. R. Humayoun, Y. Dubinsky, and T. Catarci, "A three-fold integration framework to incorporate user – centered design into agile software development," *ser. Lecture Notes in Computer Science*, vol. 6776, pp. 55-64, 2011.

[10] J. Iivari, H. Isomäki, and S. Pekkola, "The user - the great unknown of systems development: reasons, forms, challenges, experiences and intellectual contributions of user involvement," *Information Systems Journal*, vol. 20, no. 2, pp. 109-117, Mar. 2010.

[11] M. Korkala, P. Abrahamsson, and P. Kyllonen, "A Case Study on the Impact of Customer Communication on Defects in Agile Software Development.," in *AGILE 2006 (AGILE'06)*, 2006, pp. 76-88.

[12] I. Sommerville, *Software engineering*. Addison-Wesley, 2007.

[13] B. Paech and K. Kohler, "Task-driven requirements in object-oriented development," in *Doorn, Jorge. Perspectives on Software Requirements.*, Boston, MA: Kluwer Academic,Print., 2004, pp. 1-25.

[14] E. Bjarnason, K. Wnuk, and B. Regnell, "Requirements are slipping through the gaps — A case study on causes & effects of communication gaps in large-scale software development," in *2011 IEEE 19th International RE Conference*, 2011, pp. 37-46.

[15] J. E. Hallows, *The project management office toolkit*. New York; London: AMACOM, 2002, pp. 202 - 205.

[16] R. L. Daft and R. H. Lengel, "Organizational Information Requirements, Media Richness and Structural Design," Management Science, vol. 32, no. 5, pp. 554-571, May 1986