State of Practice of User-Developer Communication in Large-Scale IT Projects

Results of an Expert Interview Series

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. [Context and motivation] User participation in software development is considered to be essential for successful software systems. Especially missing direct communication between users and developers can cause various issues in large-scale IT projects.**[Question/Problem]** We want to understand current practices of user–developer communication in large-scale IT projects, the factors for, and consequences of communication gaps, and what experts suggest to prevent them. **[Principal ideas/results]**: We conducted a series of semi-structured interviews with twelve experts. The experts work on the coordination of Business and IT and describe their experiences gained in 69 large-scale IT projects. The analysis of our interviews showed that direct user–developer communication is limited and that no commonly used method for the user–developer communication in the design and implementation activity exists. **[Contribution]**: The interviews helped us to understand current practices and issues resulting from missing communication. Furthermore, we can confirm the need for a method enhancing user–developer-communication in large--scale IT projects.

Keywords: user–developer communication, collaboration, coordination issues, software development, large-scale IT projects, expert interviews

1 Introduction

User participation and involvement (UPI) are widely studied in different fields, such as information systems (IS), human--computer--interaction (HCI), and requirements engineering (RE). Many empirical studies revealed that an increase in UPI and in particular in user-developer communication (UDC) in software (SW) development increases system success [10]. The terms “user participation” and “user involvement” are often used interchangeably, but there are also publications that distinguish between them. In our study, we use the two separate definitions of (Barki and Hartwick 1994). Thus, we define user involvement as a ‘psychological state of the individual, defined as the importance and personal relevance of a system to a user’ and user participation as ‘behaviors and activities users perform in the system development process’. User participation takes place when the end user takes an active part in the development or design process together with the designer (Hope and Amdahl 2011). User-Developer Communication is a specific form of user participation and we define it as communication, evaluation, and

adfa, p. 1, 2011.

approval activities that take place between users and IS staff (Hartwick and Barki 1994), also the frequency, content and direction of that communication (Kristensson et al. 2011).

For example, Amoako-Gympah and White found a positive correlation derived from the level of communication between the users and the IS team towards user satisfaction as a measurement for system success [1]. In addition, Barki and Hartwick [2–4] studied the dependencies between the user--IS relationship on UPI and confirmed that informal and formal communication of users with the IS team and senior management significantly influences the management of a software project and the system design, but not necessarily the satisfaction with the system [3, 4]. McKeen et al. [9] investigated contingency factors for user satisfaction and found that UDC is an independent predictor for user satisfaction. There are several methods to support UPI in software development projects and an analysis of practices of proposed solutions in our previously conducted systematic mapping study on UPI showed the importance of the setup of structures to enable communication within these methods [10]. For example, several authors suggest to clarify roles of users and mediators to reduce communication barriers [11–15]. However we did find method that supports UDC in the design and implementation phase of software development.

Begier [16] mentions that it is important to keep people (e.g. users) informed and to give them timely feedback. Particularly in the design and implementation phase, Kautz [17] suggests to have weekly feedback meetings with onsite customers during the presentations of working software. Several research studies on agile methods have targeted communication problems in software development (e.g. [12, 18]).

However, we have seen from our experience as a management consultant for IT projects that a lot of large-scale IT projects still use traditional methods, i.e. the waterfall approach. Therefore, we wanted to understand current practices of large--scale IT projects, mainly with a focus on projects using traditional development methods.

We define *large-scale IT projects* as projects which at least fulfill two or more of the following characteristics: large amount of users (over 1000 users), rollout of the system in multiple countries or business units, large budget (over 1 million EUR), project duration of one year minimum (12 calendar months).

For the term *user--developer communication,* we build upon a definition of [2], and include: 'all interaction (e.g. communication, evaluation, or approval activities) that take place between the users and developers of an IT project; we also include communication interactions that are mediated through project management.'

The role of a user includes all users from the (business) organization using the new system and their managers for approval interactions [19]. The role developer includes all IT personnel, e.g. designers, architects, coders, IT managers that are involved in the software development project.

There are other studies on communication issues and structures (e.g., regular meetings or workshops) in software development [20–22]. However, none of these studies focusses on UDC in large-scale IT projects.

We conducted an interview series with experts in large-scale IT projects to find out how and how well large-scale IT projects support UDC. In particular, we are interested to answer the following research questions:

RQ 1: Do users and developers communicate in large scale IT projects?
RQ 2: What are possible organizational obstacles that prevent large--scale IT projects from implementing UDC?
RQ 3: What factors might cause communication gaps between users and developers and what are the consequences of these communication gaps?
RQ 4: What do experienced practitioners suggest to overcome the obstacles for the implementation of UDC and to eliminate the factors that cause communication gaps?

So far, the research on UDC in large-scale IT projects provides only limited empirical insights from practitioners. We do believe that it is important to consider their perspectives and knowledge on the existing communication between users and developers and why they think it is hard to implement processes that ensure effective cooperation between the two parties. This is especially essential for the design of methods to improve UDC in large-scale IT projects.

The paper is structured as follows. In Section 2, we present related work and in Section 3, we explain the research method of the interviews and the data of the interview partners. We present the results and the discussion on the state of practice of UDC in large-scale IT projects in Section 4. Further, in Section 5, we describe the threats to validity. We conclude our work and describe future areas for research in section 6.

2 Related Work

In the introduction we referenced research on the importance of UDC: Here we present other empirical studies, which explored communication in various software development projects or settings. None of the presented studies focuses on the communication from the developer to the users in large-scale IT projects, but we will compare their results to ours and discuss similarities in Section 4. An interesting study has been done by Bjarnason et al. in 2011 [20]. They empirically studied communication gaps in terms of their root causes, causes, and effects with practitioners in one large company that develops market-driven software. However, the study focuses on the communication of requirements and as the context was market-driven software development, the results do not include communication with customers, i.e. users of the software. Stapel and Schneider [23] propose an approach of how to manage knowledge on communication and information flows in global software projects. They identified poor communication as a main obstacle to successful collaboration. However, they focus on distributed development settings and not on large-scale IT projects. Marczak et al. [24] explored information flow patterns in requirement-dependent social networks. In particular, they studied communication and coordination in cross-functional teams that work on the same or on interrelated requirements. They only looked into the communication between IT personnel and did not study the communication with the users. Lastly, Gallivan and Keil [25] studied the UDC process in a software project that failed despite a high level of user involvement. They found out that communication gaps occurred because the developers were not informed about the underlying reasons of why the users did not accept the

software system. However, their results are based on only one project, and thus include insights from a limited perspective.

3 Research Method

In order to answer our research questions, we conducted a series of interviews with twelve experts in large-scale IT projects from October until December 2012. The first interview was used as a prototype interview, in order to refine the questionnaire and estimate the time frame. We conducted qualitative interviews, which is the most important data gathering tool in qualitative research and is extensively used in IS research [26]. The interviews were semi-structured, which means they were based on a questionnaire (see Appendix), but we improvised and changed the order of questions whenever the discussion moved in another direction, as recommended by [26]. In the following, we describe the identification of the experts, the interview process, and the data analysis.

Identification of Experts. In order to ensure the right target group for our interviews, we developed a role description for people working on the coordination of Business and IT (Table 1). We believe that people fulfilling this description have a project management perspective and thus are knowledgeable about existing communication structures between developers and users. In addition, we wanted to ensure that our interview partners were experts in large-scale IT projects with experience in one or more large-scale IT projects. As we wanted to support projects using traditional methods, we searched for experts who ideally have been involved in projects not using/applying agile methods, but did not limit our search to those. As consultants are typically not involved in the whole IT project timeline, we set a minimum time of three months of participation. We used these role descriptions together with some information about our research area and the goals of the interviews to contact possible interview partners. We mainly used already existing relationships of all the authors to contact possible experts.

Table 1. Role description

Coordinator between Business and IT
▪ Involved in more than 1 large-scale IT project ▪ Ideally experiences in projects with no usage of agile development methods ▪ Involved for at least 3 months in the projects (for consultants) ▪ Person (internal or consultant) who had a leading role in the development/ implementation/customizing in a large-scale IT project and was involved in discussions with users during the project or in change request management after go-live OR who had a leading role in the requirements analysis, concept development, or project management in a large-scale IT project and was involved in defining requirements and in discussions with developers during the project and/or involved in the change request process after go-live

Overall, we could attain twelve experts for our interview series. The educational background of the interview partners is very widespread (Table 2). Furthermore, the study background covered seven different areas, with half of them in IT- related subjects (4 in Computer Science and 2 in Information Technology).

Table 2. Overview of base data of experts

No.	Role in Company	Perspective (Industry)	Educational Background	# of Projects
1	Project manager	Internal IT (Pharma)	Mathematics	15
2	Business project manager	Management consulting	Business Administration and Engineering	6
3	Developer, architect, requirements engineer	IT consulting	Computer Science	3
4	Business project manager	Management consulting	Mechanical Engineering	3
5	Developer, head of research department	IT consulting	Computer Science	5
6	IT project manager	IT consulting	Information Technology	6
7	Business project manager	Internal IT (Insurance)	Mathematics	2
8	Head of IT Strategy	Internal IT (Public Sector)	Computer Science	3
9	IT project manager	IT consulting	Computer Science	4
10	CEO	Management Consultant and Software Company	Physics	14
11	IT project manager	IT consulting	Apprenticeship as Bank Clerk	5
12	Head of IT Strategy	Internal IT (Insurance)	Information Technology	3
			Sum / Average	**69 /** 6
			Min -- Max	2 -- 15

Seven experts are employed by IT or management consultancies, four experts work in internal IT departments of large organizations, and one expert works for software providers. If we consider the current roles of the experts within their companies, we can see that all experts have a leading role, which enables them to have a broad overview of IT projects. We also asked the interview partners in how many large-scale IT projects they were involved. On average, the interview partners were involved in six large-scale IT projects (minimum two projects and maximum 15 projects) throughout their carriers in various roles (e.g. developer, project manager, architect, requirement engineer, consultant, quality manager), which ensures a wide expertise of all of them.

In order to get an overview of the previous experience of the experts and to understand what large-scale IT projects are performed in practice, we asked the interview partners about the main characteristics of their projects respectively *System Type, Development Type, Industry, Project Length, Project Volume, Number of Users, Rollout in Countries/Business Units, Development Method, Role/Task*. Even though the experts could not name all characteristics of each project (also for confidentiality reasons), we were able to record data of 42 projects (see Appendix Figure 1).

Interview Process. Four interviews were done in person; the other eight interviews were conducted via telephone. The average time for one interview was 90 min, with a minimum of 44 minutes and a maximum of 125 minutes. In total, we collected about 18 hours of interview time. In the interviews, we explained the purpose of our research on UDC. We asked the interview partners about their experience in large-scale IT projects (see questionnaire in Appendix). With regard to our research questions, we did a mapping of the interview questions to the research questions. RQ 1 corresponds to question 6. RQ 2 and 3 correspond to question 7 and RQ 4 to question 8. Within the interviews

we used different terms and formulations than in the RQs, in order to ensure understandability for our practice experts.

Data Analysis. All interviews were recorded with the permission of the interviewees and transcribed for analysis purposes. Three experts reviewed their transcripts and all experts validated the derived results, i.e. reviewed them and approved for publication. We coded the interviews which helped us in the analysis of the results [27]. We built a code tree based on our research questions with descriptive codes and extended and reorganized the code tree in two cycles of coding [27]. We used the software MaxQDA and therefore were able to also do cross-interview or cross-code analysis (e.g. between the factors for communication gaps and ideas to overcome the factors). For the representation we use tables, which show the descriptive codes and the corresponding number of occurrences in the interviews (see Section 4). One occurrence means an expert did describe something in an interview that we mapped to a descriptive code. Thus it is possible to have two occurrences for one research question in one interview. For example, if in one interview an expert described the factors for communication gaps 'Lack of motivation of developers or users' and 'Lack of common language between Business and IT', we counted one occurrence for each of the descriptive codes. However, we ensured that each descriptive code got a maximum of one occurrence per interview. Thus, it is not possible to have more than twelve occurrences per descriptive code.

4 Results and Discussion

In this section, we describe the interview results on current communication structures (e.g., meetings, reports, workshops) in large-scale IT projects. We use tables, which show the descriptive codes and the corresponding number of occurrences in the interviews (for detailed explanation see Section 3). Within each subsection, we first present the results and the table and then discuss and compare them to the existing literature. To answer our research questions, we analyzed whether the interview partners experienced UDC in large-scale IT projects (section 4.1.). We report on organizational obstacles (section 4.2.), factors for communication gaps, and consequences of these communication gaps within the IT projects (section 4.3.). And, we describe the experts' ideas to overcome these obstacles and factors for communication gaps (section 4.4.).

4.1. Existence of UDC in Large-Scale IT Projects (RQ 1)

To understand the current practice of UDC in large-scale IT projects, we asked all interview partners, what communication took place within their projects. We wanted to understand UDC on a detailed level, thus asked exactly who communicated with whom in the project. Overall, only three experts reported of projects where communication between users and software coders (i.e. developers) took place. However, two of these three experts also participated in projects where no direct communication between those parties existed. Hence, eleven experts told us about large-scale IT projects, in which they did not experience direct communication between software coders and users (Table 4). In total, less than one fifths of all 69 projects our experts were involved in had any communication between users and developers.

Nevertheless, some projects had other forms of UDC, such as: communication between the IT consultant and the users, communication between the architect and the users, or communication between the requirements engineer and the expert users (not users, but rather business personnel with broad context knowledge or a management role). Even though our analysis of existing methods for UPI in the systematic mapping study [10] indicated that methods affect all activities of software development, we learnt from our interview partner that in practice most of the communication is done either in the early or the late activities of software development (i.e. in specification or acceptance).

Table 4. Existence of direct communication between developers and users

Existence of UDC (Descriptive Code)		**# of Int.** [1]
Communication between software coders (i.e. developers) and users		3
No communication between software coders (i.e. developers) and users		11
Other forms of communication with users		
	Communication between IT consultant and users	3
	Communication between architect and users	2
	Communication between requirements engineer and expert user	2

Based on the experiences of our experts, we can conclude that direct communication between developers and users does not exist in most large-scale IT projects. This is in contrast to Chang et al.'s results [5], who found that the presence of mutual influence among IT staff and users, which enables open and direct communication and coordination, is significantly associated with project performance. However, their context was not within large-scale IT projects. The reported setup of communication between requirements engineers and expert users is in line with Kanungo and Bagchi [6], as they suggest moving user participation upstream in the implementation process and using representatives of user groups. The finding that most of the communication is done either in the early or the late activities of software development shows a lack of communication in the middle of the development, i.e. in the design and implementation activity. Even though, there are suggested methods in literature, e.g. Kautz [17] and Korkola [15] suggest to have weekly feedback meetings with onsite customers during working software presentations or at least mid-iteration communication with users, our findings show that the implementation of such methods is limited in practice.

4.2 Organizational Obstacles for UDC (RQ 2)

We identified four obstacles whereof three concern the users or access to them (Table 5). In total, we did discuss the topic of organizational obstacles with half of the experts; the other experts did not mention any organizational obstacles. Firstly, two experts mentioned that users are not a homogeneous group, but different user groups or business units with often different opinions and organizational power within a company. In such cases, developers (and other IT personnel) face an additional challenge, as they need to mediate between these groups. Secondly, it seems to be hard to find user representatives with the right qualification and knowledge for an IT project. We think, this can be ex-

[1] Number of interviewees that mentioned an experience mapped to descriptive code

plained by the fact that knowledgeable key users are very important for the business operations and thus will not be released to fulfill tasks within IT projects. Thirdly, one expert mentioned that in several projects the real users are not defined during the project, thus the developers (and other IT personnel) cannot access them. Fourthly, one expert reported that no mediators were available to establish and uphold the relationship between the users and the developers.

Table 5. Organizational obstacles for implementing communication with users

ID	Organizational Obstacles (Descriptive Code)	# of Int.
O1	Different opinions between user groups	2
O2	Get the right user representatives for large-scale projects	2
O3	No access to users/users unknown	1
O4	Lack of local mediators	1

The obstacles O1 and O2 correspond with the findings of Bjarnason et al. [20], who also identified scale effects through complex products and large organization. In addition, they describe gaps between roles over time through distributed environment as root causes for communication gaps. Even though they studied a different setup without direct contact to users, these obstacles also seem to be present for UDC. Obstacle O4 is supported by the findings of Marczak et al., who studied communication and coordination in cross-functional teams that work on the same or interrelated requirements [24]. They found out that the power of information flows lies with a few key members who control information flows between dependent networks. Our findings indicate that this is also true for UDC.

4.3. Factors for and Consequences Caused by Communication Gaps (RQ 3)

We identified three factors for communication gaps and four consequences caused by communication gaps (Table 6). Common factors for communication gaps are 'lack of motivation of either the users or the developers', as well as the 'lack of a common language between the business and IT side'. Another factor, which is somehow related to both of the other factors, is lack of appreciation between these two sides. The consequences most frequently named among interviewees is the misunderstanding of requirements, i.e. developers either interpret requirements in a wrong way or users do not specify requirements on a detailed level and are later surprised by the results. This also often leads to the need of ad-hoc changes or, as one expert named it, a 'scope creep' during implementation. In addition, increased implementation cost or test effort were named as consequences of communication gaps.

Table 6. Factors for and consequences caused by communication gaps

ID	Factors for communication gaps (Descriptive Code)	# of Int.
F1	Lack of motivation of developers or users	4
F2	Lack of common language between Business and IT	4
F3	Lack of appreciation between Business and IT	1
	Consequences caused by Communication Gaps (Descriptive Code)	**# of Int.**
C1	Misunderstanding of requirements	8

C2	Ad-hoc changes required due to unclear requirements	3
C3	Increased implementation cost	3
C4	Increased test effort due to rework	1

The results of RQ 3 show that the consequences are severe as misunderstandings and ad-hoc changes have an impact on cost and schedule of the project. The factor F1 is similar to Bjarnason et al.'s [20] identified effect of "low motivation to contribute to requirements work" and F2 is a commonly known issue in IT projects. However, the factor F3 of missing appreciation has not been described so far and is also interesting, as the required actions to improve appreciation between IT and Business are different from overcoming barriers of a common domain language. The identified consequences C1 and C2" are in line with Bjarnason et al's effect [20] described as "problems with the system requirements specification". C3 and C4 are similar to their effect "wasted effort". However, it is quite interesting that our results show that the experts stated a clear connection between communication gaps and increased implementation costs and a higher test effort. In addition, the consequences C1 to C4 correspond to the named benefits of UPI [10], such as improved quality due to more precise requirements and the prevention of expensive features.

4.4. Ideas to Overcome Obstacles for the Implementation of UDC and Factors for Communication Gaps (RQ 4)

In total, the experts suggested twelve different approaches to overcome factors for communication gaps or obstacles. We classified these approaches into three categories, user-centered approaches, developer-centered approaches, and organizational approaches. We mapped them in our analysis phase to the addressed factors for communication gaps and organizational obstacles wherever possible and identified similar approaches from the literature (Table 7). The user-centered approaches are ideas that include the involvement of the user. The second category clusters ideas that have to be realized by the developer. The third category of organizational approaches is for ideas that need to be considered in the setup of the project organization and management.

Table 7. Ideas to overcome obstacles or factors for communication gaps

Category	Ideas (Descriptive Code)	# of Int	Literature	ID	Addressed Factor/Obstacle
User-centered approaches	Presentation of (UI) prototypes or proof of concepts to users	3	[14, 28, 29]	O2	Get the right user representatives for large-scale projects
	House tours in different business units with running SW	1	[17, 30]	F2	Lack of common language between business and IT
	Description of added value to users to increase acceptance	1	n/a		
	Incentive system for the participation of business users	1	[31]	F1 O2	Lack of motivation of developers or users Get the right user representatives for large-scale projects
	Involvement of users in the organization of rollout and change management	1	n/a	O2	Get the right user representatives for large-scale projects
Developer-	Developers must mediate between different user groups	2	[13]	O1	Different opinions between user groups

centered approaches				O4 F2 F3	Lack of local mediators Lack of common language between Business and IT Lack of appreciation between Business and IT
	End-to-end feature responsibility of developers	1	n/a	F2	Lack of common language between Business and IT
	Developer writes informal description of how to implement requirements.	1	n/a		
	Obligation to justify all technical decisions with functional need	1	n/a		
Organizational approaches	Usage of test data early in project	2	[32]		n/a
	Agile methods e.g. frequent review meetings	2	e.g. [17, 33]		
	Definition of usability guidelines to avoid detailed UI discussions	1	n/a		

In the first category of user-centered approaches, five ideas were named.

One idea is to show the users prototypes (often called ‘proof of concept’ by the experts). One expert described a successful project: the software was very complex, therefore the project members wrote down all requirements in large workshops with about 50 users and then invited two vendors to build up prototypes as a ‘proof of concept’ before the actual design and implementation activity began. The users were highly involved in this activity, as the vendors presented the status of the prototype in regular meetings to them. At the end of the proof of concept activity, a prototype, implementing about 80% of the functionality, had been built and was aligned with the users. The vendor selected for implementation could proceed with implementing the rest of the requirements, integrating the prototypes into the system’s landscape, and building up the data structures. Even though this is a promising approach, the expert mentioned that it will be hard to implement in large-scale IT projects such as an ERP implementation, because those systems’ functionality is too wide for a prototype approach. Nevertheless, two other experts suggested showing users mockups or even integrate users as beta customers within the software development by showing them running prototypes. In general, this idea of using prototypes is not new and has been described in the literature, e.g.[14, 28, 29]. However, the detailed description of how such an approach was used within a real-life IT project can be helpful for the research community.

Another suggested approach that is similar to the prototype approach described above is to do house tours with running software. The difference to the proof of concept approach is that after about half of the actual implementation time, the project team presents the running software in different business units to different users. This approach allows small changes of the system based on user feedback and it ensures an early change and expectation management with the users. A similar approach has been described by [17, 30]. They call it “road shows” and suggest having onsite users conducting them with other users.

One approach in response to the factor ‘lack of common language between business and IT’ was to explain the added value of the system to the users. The expert suggests doing that with posters, result descriptions, and several meetings with the users.

To include users in the rollout and change management planning was also named by an expert. According to the expert, this leads to a higher integration of users in the pro-

ject. For these two suggestions, we could not identify an approach from the literature, thus these are particularly interesting findings for the design of a new method.

The last suggestion in that category has been for years in the head of one of our interview partners, namely to create an incentive system for the participation of business users. The expert wants to overcome the factor 'lack of motivation of users' and the obstacle 'get the right user representatives for large-scale projects'. One issue, in the opinion of the expert, is that users are not rewarded either through promotions or higher wages for their work in IT projects in addition to their usual daily work. This lack of appreciation leads to a low interest and thus low involvement of the user. A similar idea has been presented by Finck et al. [31] They suggested an incentive system for the software evolution activity, i.e. after the first rollout of a system.

In the category of developer–centered approaches, four ideas were named.

Especially in response to the obstacle 'different opinions between user groups," two experts recommended that developers need to mediate between different user groups. As different user groups (e.g. the finance and the marketing department) often have different opinions, the developers need to solve their communication gap and dissolve their disagreement.

In addition, one interview partner referred to the factor 'lack of appreciation between Business and IT' and 'lack of common language between Business and IT' by explaining: "Most (non IT) users do not think in structures…thus the IT personnel need to learn to talk in examples to explain their structure, even though it is not relevant to them." Therefore, this expert suggests always having someone in the project, who has experience with the to-be-implemented business domain. This person can then fulfill the mediator role. In general, the idea to clarify roles and mediators is described in the literature, e.g. [13], but to assign/fill this role to/with a developer is a new suggestion. With reference to the factor 'lack of common language between Business and IT', one expert suggested to ensure end-to-end feature responsibility for each developer. That means, you do not need a developer who is responsible for one technical cross over area, e.g. database or UI, you rather need a developer who is responsible for the implementation of one use case, including the UI, the business logic, the database, and the interfaces.

A similar approach is to oblige the developer to write an informal description of how to implement a given requirement so the users should be able to read and understand information related to implementation. Before the implementation starts, this informal description must be aligned with the users. We think this also helps to mitigate all the above mentioned four consequences of communication gaps. In order to mitigate the lack of a common language, one interesting approach is the obligation for developers to justify all technical decisions with a functional need. For example, the need for another database can only be justified with a higher service level for the business unit, but not out of narcissistic technical preferences of a developer. The last three developer-centered approaches have so far not been described in the UPI literature. Thus, it is important to include these suggestion is future work of methods to improve UDC.

In the category of organizational approaches three ideas were named.

The usage of test data very early in the software development process is supposed to give the users a possibility to challenge the logic and the quality of the system. One expert suggested using extreme test data to provoke situations where complications can occur. Another expert suggested having usability tests with real data as early as possible within a large-scale IT project, which has also been suggested in [32].

Another suggested approach was, to use agile methods, e.g. have weekly or monthly meetings (often called sprint meetings) together with user representatives. Even though this expert suggested this approach, he also reported that those meetings had not been a success, which he attributes to the too finely-grained level, i.e. on a bug tracker level. This was too detailed for the users and they lost attention after two minutes. Furthermore, these meetings had been held as a telephone conference which, according to the expert, is not the ideal setting. Agile methods including a high level of feedback towards the users have been described extensively in the literature, e.g., [17, 33].

In addition, one expert mentioned that it is not only important to involve users by offering workshops or by showing prototypes to them, but also to ensure clear guidelines for the user interface. This is particularly important in terms of the user interface, as several unnecessary discussions about screen details occur in meetings with users. The expert also mentioned that if these guidelines are missing this can have high cost implications for the project.

Overall, we can conclude that the experts' ideas try to overcome all factors for communication gaps (F1 – F3) and the organizational obstacles (O1, O2, O4), except obstacle O3, namely the "lack of access to users". Nevertheless, the experts did not report of a successful, sustainable solution to overcome the communication gaps in large-scale IT projects and in particular in the design and implementation phase.

5 Threats to Validity

We analyzed threats to validity based on the scheme suggested from Runeson [34].

Construct validity – as described in the research method section, the interviews were semi--structured thus interviewees and interviewer could influence the direction of the discussion, which sometimes led to the fact that we did not pose all questions of our interview guideline explicitly. Furthermore, eight interviews were conducted via telephone, which prevents visual cues and sometimes limited the understanding. We mitigated that threat through the recording of all interviews. This also enabled us to rewind for the transcripts in the case of poor acoustic reception.

Internal validity – we relied on our personal relationships for the identification of experts, this can be a threat to internal validity, as three of the experts knew the interviewer before the interviews and therefore they might be biased. However, the majority of the experts did not know the interviewer.

External validity – a possible threat to external validity is that we only interviewed twelve experts. Nevertheless, the experts' backgrounds were very widespread and they all had been involved in a minimum of two large-scale IT projects. Therefore, we are confident that our results show a broad overview of communication structures in large-scale IT projects and can be transferred to other projects outside of the experiences of our interviewees.

Reliability – The interviews as well as the coding of the interviews were conducted by one person. On the one hand, this ensured the consistency of the interviews and their analysis. On the other hand, it can also be a threat to the reliability, as another researcher could interpret the results in another direction.

6 Conclusion

In this paper, we reported on the results of an interview series with experienced practitioners in large-scale IT projects. We conducted twelve semi-structured interviews, transcribed all interviews and coded them with descriptive codes based on our research questions. Our experts described experiences from 69 large-scale IT projects, which ensure widespread experience. In the context of our larger research on UDC in large-scale IT projects, we wanted to determine how and how well large-scale IT projects support UDC.

With regard to current communication structures in large-scale IT projects, the results of the study indicate that *direct communication between developers and users does not exist in most large-scale IT projects*. The experts describe some setups for communication with the users, e.g. communication between IT consultant and users, but none of them seems to focus on our research target the design and implementation activity.

The identified obstacles for implementation and factors for communication gaps seem to be in line with the literature [20, 35], e.g. lack of motivation of user or developer or a lack of a common language of Business and IT. However, an interesting result is that the experts stated a clear connection between communication gaps and increased implementation costs and a higher test effort.

We classified the ideas from experts to overcome the obstacles in *user-centered approaches*, e.g. show user prototypes, *developer-centered approaches*, e.g. developers must mediate between different user groups and *organizational approaches*, e.g. use test data early in the project. Some of the suggestions have also been described in the literature, however the detailed descriptions of which setup was successful in large-scale IT projects and the developer-centered approaches are important findings for our future work. The experts did not report on a successful, sustainable solution to overcome the communication gaps in large-scale IT projects and in particular to improve UDC in the design and implementation activity.

In our future work, we plan to detail our method to support UDC in large-scale IT projects. We already published a first proposal and a descriptive classification of user-relevant decisions in two other papers [36, 37] Furthermore, we plan to evaluate the implementation feasibility as well as measure the benefits of the method in a case study in a large-scale IT project.

Acknowledgement. We would like to thank all experts for their time and support of this research.

References

1. Amoako-Gyampah, K., White, K.: User involvement and user satisfaction: An exploratory contingency model. Inf. Manag. 25, 25–33 (1993).
2. Barki, H., Hartwick, J.: Measuring User Participation, User Involvement, and User Attitude. MIS Q. 18, 59 (1994).
3. Hartwick, J., Barki, H.: Communication as a dimension of user participation, (2001).
4. Hartwick, J., Barki, H.: Delineating the dimensions of user participation: A replication and extension. Rev. Lit. Arts Am. (1997).

5. Chang, K., Shin, T., Klein, G., Jiang, J.J., Sheu, T.S.: User commitment and collaboration: Motivational antecedents and project performance. Inf. Softw. Technol. 52, 672–679 (2010).
6. Kanungo, S., Bagchi, S.: Understanding User Participation and Involvement in ERP Use. J. Manag. Res. 1, 47–64 (2000).
7. Kristensson, P., Gustafsson, A., Witell, L.: Collaboration with Customers -- Understanding the Effect of Customer--Company Interaction in New Product Development. 2011 44th Hawaii International Conference on System Sciences. pp. 1–9. IEEE (2011).
8. Kujala, S., Kauppinen, M., Lehtola, L., Kojo, T.: The Role of User Involvement in Requirements Quality and Project Success. 13th IEEE Int. Conf. Requir. Eng. 75–84 (2005).
9. McKeen, J., Guimaraes, T., Wetherbe, J.: The Relationship between User Participation and User Satisfaction: An Investigation of Four Contingency Factors. MIS Q. 18, 427–451 (1994).
10. Abelein, U., Paech, B.: Understanding the Influence of User Participation and Involvement on System Success -- a Systematic Mapping Study. Empir. Softw. Eng. in press (2013).
11. Amoako--Gyampah, K., White, K.: When is user involvement not user involvement? Inf. Strateg. Exec. J. 13, 40 – 45 (1997).
12. Hope, K., Amdahl, E.: Configuring designers? Using one agile project management methodology to achieve user participation. New Technol. Work Employ. 26, 54–67 (2011).
13. Eckhardt, A.: Lost in Translation ?! – The Need for a Boundary Spanner between Business and IT. SIGMIS--CPR'10, May 20–22, 2010, Vancouver, BC, Canada. pp. 75–82 (2010).
14. Humayoun, S., Dubinsky, Y., Catarci, T.: A Three--Fold Integration Framework to Incorporate User – Centered Design into Agile Software Development. Hum. Centered Des. HCII. 6776, 55–64 (2011).
15. Korkala, M., Abrahamsson, P., Kyllönen, P.: A Case Study on the Impact of Customer Communication on Defects in Agile Software Development . In: Abrahamsson, P. and Kyllonen, P. (eds.) AGILE 2006 (AGILE'06). pp. 76–88. IEEE (2006).
16. Begier, B.: Evolutionally Improved Quality of Intelligent Systems Following Their Users ' Point of View. Advances in Intelligent Information and Database Systems. pp. 191–203 (2010).
17. Kautz, K.: Investigating the design process: participatory design in agile software development. Inf. Technol. People. 24, 217–235 (2011).
18. Takats, A., Brewer, N.: Improving Communication between Customers and Developers. Agil. Dev. Conf. Database Conf. 0, 243–252 (2005).
19. Carmel, E., Whitaker, R.D., George, J.F.: PD and joint application design: a transatlantic comparison. Commun. ACM. 36, 40–48 (1993).
20. Bjarnason, E., Wnuk, K., Regnell, B.: Requirements are slipping through the gaps — A case study on causes & effects of communication gaps in large--scale software development. 2011 IEEE 19th International Requirements Engineering Conference. pp. 37–46. IEEE (2011).
21. Stapel, K., Knauss, E., Schneider, K., Zazworka, N.: FLOW Mapping: Planning and Managing Communication in Distributed Teams. 2011 IEEE Sixth Int. Conf. Glob. Softw. Eng. 190–199 (2011).
22. Marczak, S., Kwan, I., Damian, D.: Social Networks in the Study of Collaboration in Global Software Teams. 7–8 (2007).
23. Stapel, K., Schneider, K.: Managing knowledge on communication and information flow in global software projects. Expert Syst. 00, n/a–n/a (2012).
24. Marczak, S., Damian, D., Stege, U., Schröter, A.: Information Brokers in Requirement--Dependency Social Networks. 2008 16th IEEE Int. Requir. Eng. Conf. 53–62 (2008).
25. Gallivan, M.J., Keil, M.: The user--developer communication process: a critical case study. Inf. Syst. J. 13, 37–68 (2003).
26. Myers, M.D., Newman, M.: The qualitative interview in IS research: Examining the craft. Inf. Organ. 17, 2–26 (2007).
27. Saldana, J.: The Coding Manual for Qualitative Researchers (Google eBook). (2009).

28. Cohen, S., Dori, D., Haan, U. De: A Software System Development Life Cycle Model for Improved Stakeholders ' Communication and Collaboration. Int. J. Comput. Commun. Control. 5, 20–41 (2010).
29. Dean, D., Lee, J., Pendergast, M., Hickey, A., Nunamaker, J. ay: Enabling the Effective Involvement of Multiple Users : Methods and Tools for Collaborative Software Engineering. J. Manag. Inf. Syst. 14, 179 –222 (1998).
30. Martin, A., Biddle, R., Noble, J.: An Ideal Customer: A Grounded Theory of Requirements Elicitation, Communication and Acceptance on Agile Projects. Agile software development : current research and future directions. pp. 111–141. Springer, Berlin (2010).
31. Finck, M., Gumm, D., Pape, B.: Using Groupware for Mediated Feedback. Proceedings of the eighth conference Biennial Participatory Design Conference 2004: Artful integration: interwearing media, materials and practices -- volume 2, July 27 -- July 7, 2004, Toronto, Canada. (2004).
32. Teixeira, L., Saavedra, V., Ferreira, C., Santos, B.: Using Participatory Design in a Health Information System. Conf. Proc. Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. 2011, 5339–42 (2011).
33. Korkala, M., Pikkarainen, M., Conboy, K.: Combining Agile and Traditional: Customer Communication in Distributed Environment. In: Šmite, D., Moe, N.B., and Ågerfalk, P.J. (eds.) Agility Across Time and Space. pp. 201–216. Springer Berlin Heidelberg, Berlin, Heidelberg (2010).
34. Runeson, P., Host, M., Rainer, A., Regnell, B.: Case Study Research in Software Engineering. Wiley--Blackwell (2012).
35. Harris, M., Weistroffer, H.: A New Look at the Relationship between User Involvement in Systems Development and System Success Development and System Success. Commun. Assoc. Inf. Syst. 24, 739–756 (2009).
36. Abelein, U., Paech, B.: A proposal for enhancing user--developer communication in large IT projects. Proceedings of the 5th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE 2012) at the ICSE 2012 Zurich, June 2nd (2012).
37. Abelein, U., Paech, B.: A Descriptive Classification for End User --Relevant Decisions of Large--Scale IT Projects. Cooperative and Human Aspects of Software Engineering (CHASE), 2013 6th International Workshop on. p. accepted (2013).

7 Appendix

Interview questionnaire

1. What is your role in your company? What is your educational background?
2. How many large IT projects (either large amount of users, multiple countries or business units involved, large budget, project duration minimum of 1 year, e.g. ERP implementation) have you been involved in?
3. What were the main characteristics of these projects (type of system, project length, amount of users)?
4. What was your role and what were your tasks within these projects?
5. Would you classify yourself on the IT or on the Business side?
6. Was there communication between users and developers of the project? If yes in what setup did the communication take place? In what SW activities of the project did the communication take place?
7. Did you experience any issues/consequences in these projects that might be caused by communication gaps? If yes, please specify the issues. In what SW activities did the issues occur?
8. What would you do to prevent these issues in your next project?

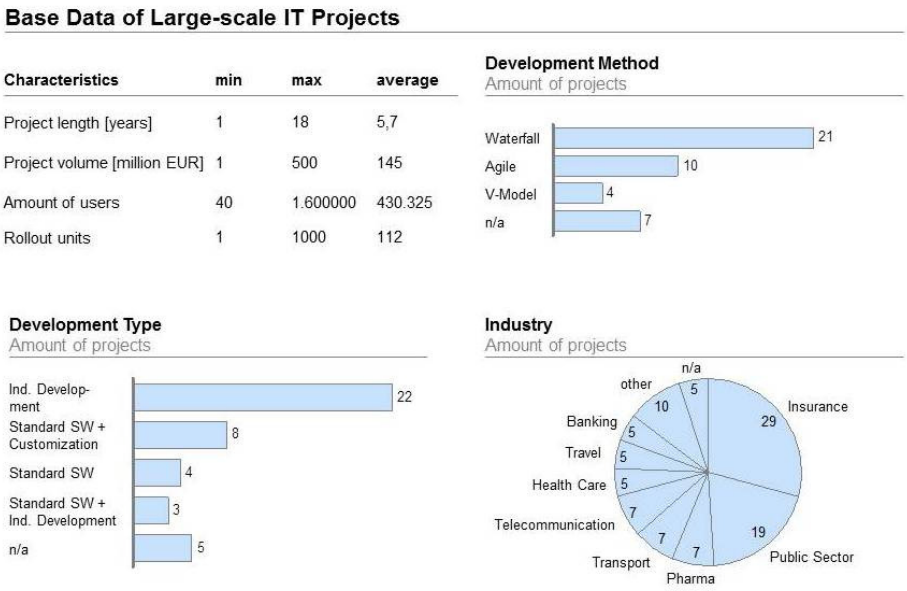

Fig. 1. Base Data of Large--Scale IT Projects