The Challenges of Distributed Software Engineering and Requirements Engineering: Results of an Online Survey

Technical Report SWEHD-TR-2007-03

Timea Illes-Seifert, Universität Heidelberg, Arbeitsgruppe Software Systeme
Andrea Herrmann, Universität Heidelberg, Arbeitsgruppe Software Systeme
Barbara Paech, Universität Heidelberg, Arbeitsgruppe Software Systeme

Version 1.0
September, 2007

Eine Publikation der Arbeitsgruppe Software Engineering

History

Version	Date	Change history
V.1.0	September 2007	Initial version

The „Software Engineering“ Group is part
Of the Institute of Computer Science of the
Ruprecht-Karls-Universität Heidelberg.
This group is led by
Prof. Dr. Barbara Paech.

Institut für Informatik
Neuenheimer Feld 348
69120 Heidelberg
paech@informatik.uni-heidelberg.de
http://www-swe.informatik.uni-heidelberg.de/

Copyright © Arbeitsgruppe Software Systems Engineering Heidelberg

The Challenges of Distributed Software Engineering and Requirements Engineering: Results of an Online Survey

Timea Illes-Seifert, Andrea Herrmann, Barbara Paech
Universität Heidelberg, Im Neuenheimer Feld 326, 69120
{illes, herrmann, paech}@informatik.uni-heidelberg.de

Abstract

Growing globalization and increasing complexity of software lead to international and national collaboration of geographically distributed organizations, sites and persons. Therefore, it becomes more important to understand and to know how to optimize distributed software development. Thus, we performed a survey among professionals on their experiences with distributed software development. We present an evaluation of 744 questionnaires, with a special focus on requirements engineering. The survey results show that a variety of human and process-related aspects are important for distributed software development. They furthermore emphasize the importance of communication in requirements engineering: Communication, particularly face-to-face meetings, represents the most frequently mentioned solution to diverse problems. Similar results were found before, but this survey supports them with a high quantity of data.

1 Introduction

The trend towards sub-contracting, outsourcing, and off-shoring, as well as the collaboration with partner organizations or within an organization at different locations (nationally and internationally) requires the use of knowledge and resources distributed over multiple locations. Communication, as the process of knowledge exchange, is therefore an important issue for software development teams [2] - even when they are not distributed: „Software Engineering is inherently a team-based activity“ [1] and thus implies knowledge exchange among its members. In the case of distributed projects, communication becomes even more important [7], [8], [19]. Existing research indicates that means of communication, such as phones, mobile devices, email, or video conferencing equipment cannot fully substitute face-to-face meetings and demand for instance communication by traveling [4] or “get to know” meetings [6]. Although tool support and processes which support collaboration cannot guarantee a good software engineering result, they are considered to be necessary prerequisites. Requirements on such tools supporting distributed software engineering are discussed in [8] and requirements on distributed processes in [3]. A tool for requirements prioritization by “non-co-located experts” is presented in [13] and a process for distributed requirements prioritization in [21]. First studies investigated lessons learned from distributed software development [20] and distributed design [7].

As their conclusions however are based only on a few cases, we performed a quantitative online survey among software engineering professionals with the goal to investigate the state of the practice in distributed projects, including distributed requirements engineering. Particularly, we used an online survey in order to reach a high number of participants and consequently to derive statistically significant results.

Copyright © Arbeitsgruppe Software Systems Engineering Heidelberg

High participation in our survey indicates the practical significance of distributed software development (744 participants within 3 weeks). Moreover, the high degree of experience with distributed software projects among the participants underlines the study's representativeness.

In this technical report, we present the first results of the online survey, particularly we focus on Requirements Engineering. Further analyses will comprise detailed evaluations of other phases (design, implementation and test) as well as detailed correlation analyses of the data gathered in our study.

The questionnaire has been developed as conjoint work with Michael Geisser and Tobias Hildebrand, University Mannheim. Initial results of this study have been presented at the GREW conference (Global Requirements Engineering) [24] and published in [25].

The remainder of the report is organized as follows: Section 2 provides a description of the survey, while section 3 presents characteristics of the participants, of the distributed processes and of the tool usage as indicated by the respondents. Section 4 describes the analyses of participants´ responses related to challenges of distributed software development as well as issues specific to distributed requirements engineering (RE). Additionally, successful solutions to the issues mentioned by the participants are presented. Overall conclusions and future work are provided in Section 5.

2 Methodology and Study Design

Distributed Software Development. In our study, we define distributed software development as follows: "All or at least some participants of a software project predominantly use distributed technologies for team communication (e.g. because this is not possible otherwise due to geographical distance)" [12].

Questionnaire Design. We designed our survey questionnaire and the categories for the coding of answers to open questions by applying the MOQARE/misuse case principle [10], [11]. We first defined important quality goals of each – even intermediate – product, i.e. of the requirements specification, design, code, and test results during distributed software development. This quality was measured by quality attributes which were specific for each product. As the quality goal of the process we defined the efficiency in the creation of these products. Then, we identified misuse cases, which can possibly happen during the process of distributed software development and which endanger the goals mentioned above. Misuse cases describe scenarios which must be avoided. Discussing such unwanted events and the countermeasures that can detect, prevent or mitigate them, usually helps to complement requirements. In the next step, we identified such countermeasures which here were requirements for processes and tools used in the development of distributed software. We identified misuse cases and countermeasures for the different phases of the software development: for requirements engineering, architectural design, coding and testing. An example is the misuse case "The requirements specification is ambiguous because different terminologies and notations are used." Important countermeasures for this misuse case would be to use a glossary and to define a common notation. Many misuse cases could occur the same way in every activity and were classified as "general problems". For two reasons, we did not include our 137 misuse cases in the questionnaire. Firstly, this would demand too much time of the survey participants to comment on all of them, and secondly, pre-defining a list of misuse cases would focus the answers on these particular ones, without guaranteeing that the list contains the most relevant ones as experienced by the

Copyright © Arbeitsgruppe Software Systems Engineering Heidelberg

participants. Instead, the misuse cases were the basis for coding the answers to the open questions during data analysis.

The questionnaire consists of two parts. The first part contains questions on the respondent's experience with distributed software development. Particularly, we asked about the roles of the respondent, the phases which have been performed in a distributed way, as well as the technology used for communication and information sharing in distributed projects. The analysis of the respondent's answers to questions covered by the first part of the questionnaire is presented in section 3. The second part of the questionnaire addresses problems that occur in distributed projects, their causes and their solutions. This second part consists of four sets of questions. In the first set, respondents were asked about general problems in distributed projects and their solutions (here, we proposed those nine misuse cases which apply to general problems), whereas the other three sets of the questionnaire asked open questions concerning problems and solutions specific to distributed requirements engineering, software design and coding as well as software testing. An analysis of the respondents´ answers to questions covered by the first set is presented in Section 4.1, whereas comments, misuse cases and countermeasures concerning distributed requirements engineering are presented in Section 4.2.

The resulting questionnaire was thoroughly reviewed and tailored by the authors before being published online. Criteria for reviewing were above all the comprehensibility of the questions. Another criterion was the time needed for answering the questionnaire. Since respondents are not willing to deal with lengthy questionnaires [17], we aimed at developing a questionnaire, which does not take longer than 20 minutes to be completed.

Data Collection. The final version of the survey was published online for three weeks. In order to attract many participants, we promoted the questionnaire by posting an online advertisement in the news ticker of a popular German computer journal. Additionally, we addressed the participants on the mailing list of events organized by the MFG (Media and Film Association) Baden-Württemberg, a centre of excellence for information technology and media in the southwest of Germany [16].

Data Analysis. Before performing analyses, we validated the data, analyzing the responses with respect to validity and consistency as recommended in [18]. E.g. 24 participants indicated having no experience with distributed development projects, thus, we did not consider their responses in our further analysis. Finally, there were 744 valid questionnaires. The mean time for completing the questionnaire finally was 14 minutes.

After data validation, we coded the answers [18]. Thus, answers to open questions were categorized in order to be analyzed in further steps. In questions concerning the technology used to support distributed software development, we had proposed several alternatives including email and chat. However, the participants also had the opportunity to add other technologies not mentioned in the list. Some of the respondents indicated special software packages, requiring that we had to code the information and to categorize the answers by assigning the particular software solutions to a particular underlying technology.

3 Results

In the following, general characteristics of the participants, of the distributed processes and of the tool usage as indicated by the respondents are presented.

3.1. Participants Characteristics

Copyright © Arbeitsgruppe Software Systems Engineering Heidelberg

Experience of the Participants. The participants had worked in an average of 7.5 distributed projects. This shows the high qualification of the participants, and also that distributed software development is neither an exotic, nor a newly emerging phenomenon.

Roles of the Participants. The most frequent role the participants had taken in distributed projects was the one of the developer (68%) and the software architect (57%). 39% stated they were project manager, 16% requirements engineer, 29% tester and 7% in other roles. (Double and multiple roles were indicated frequently.)

Application Domains of the Software. The participants were asked about the business domain of their customers (multiple answers were possible). Most frequently, respondents stated "software" (48%) followed by the technical sector (42% including mechanical engineering, chemistry, electrical engineering, telecommunication, and transport) and service (39% covering education, consulting, IT services). The rest were commercial sector (banking, insurance) (23%), public sector (administration, government) (19%) and others (14%).

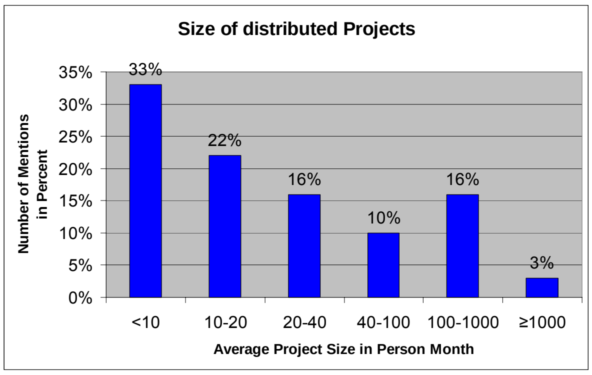
3.2. Characteristics of Distributed Processes

Size of Organizations. Software is being developed in distributed projects in big organizations as well as in small and medium sized organizations: 34% of these projects took place in organizations with more than 10.000 employees, and 38% in such with less than 100. The rest is distributed among organizations with 1000-10.000 employees (13%) and 100-999 (15%).

Size of Projects. 33% of the distributed projects had a volume of less than 10 person months and another 22% from 10 to 20. Figure 1 illustrates the average size of distributed projects as indicated by the respondents.

An average of 94 persons per project communicated via the distributed technology, and the average number of project team members was 84. It can be concluded that the distributed technology involved persons in the communication who were not project team members. On the other hand, 18% of the participants did not know the number of persons involved. This high number indicates that the distributed communication leaves some persons without an overview or "awareness" [9] about the members of the team and their activities.

Figure 1. Average Project Size in Person Month



Copyright © Arbeitsgruppe Software Systems Engineering Heidelberg

Project Phases and Roles. The project phase which had been done in a distributed way most often was implementation (92% of the participants), followed by testing (73%) and architectural design (62%). Requirements analysis (38%) and operation as well as maintenance (46%) were less frequent. This distribution among project phases is also reflected by the role distribution of the participants (see preceding section) and when being asked which roles did use the distributed technology. We want to point out that 27% of the respondents indicate that later users of the system were also involved in the development project using distributed technology for communication and information exchange.

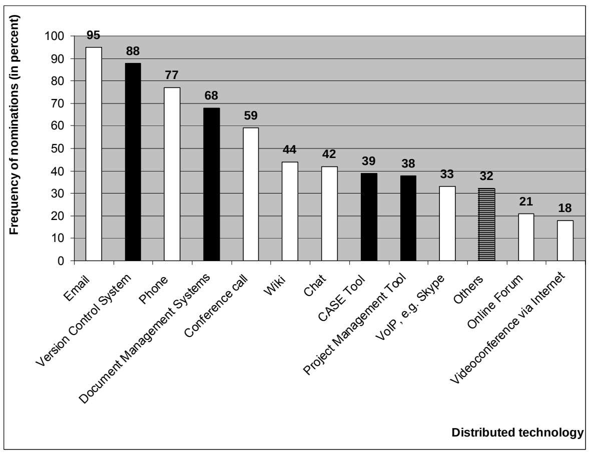
3.3. Tool Characteristics

Distributed Communication. Participants of the survey were asked to indicate which kinds of distributed technology they use for distributed communication. Email seems to be the most important tool for communication. Almost 95% of the participants indicate to use those means for asynchronous communication. The most important synchronous technologies are the phone and the conference call. 77% of the respondents indicate to use phone and 59% indicate to use conference calls in distributed projects. Other technologies used comprise video conferencing (not via internet) and remote desktops.

Distributed Information Exchange. Participants of the survey were also asked to indicate which kinds of distributed technology they use for distributed information exchange. Version control systems and document management systems referred to as the most frequently used technologies for information sharing. CASE tools and project management tools are only used by about 40% of the respondents to exchange information. Another information platform mentioned by respondents represents problem/defect management reports.

Figure 2 summarizes the results of the survey study with respect to the frequency of mentions regarding communication (white bars) and with respect to information exchange using a certain distributed technology (black bars). The finding, that email, telephone and file sharing are the most frequently used tools is consistent with results of other studies [6],[14].

Figure 2. Distributed Technology Usage



4 Analysis of Participants´ Comments, Misuse Cases and Countermeasures

In this section, we present challenges of distributed software projects and particular issues in requirements engineering drawn from our survey.

4.1. Challenges of Distributed Development

In comments and open questions concerning misuse cases, respondents mention mainly five types of challenges concerning distributed development: process barriers, cultural barriers, domain specific barriers, technical barriers and communication barriers.

Figure 3 visualizes these barriers and their corresponding specific facets by means of a fishbone diagram. The numbers in parentheses represent the number of mentions.

Process barriers are the most frequently mentioned barriers in distributed software development. 10 respondents indicate as a major problem that documented processes are not actually implemented and that responsibilities are not clearly defined (mentioned 9 times). Reasons for unclear responsibilities as mentioned by the respondents are, e.g. frequently changing responsibilities or the lack of a coordinator role. Other important process barriers represent enhanced communication needs (9) and inappropriate processes (8). A main reason for increased communication is reported in cases with incomplete documentation. Inadequate processes mainly result from the use of "standard" processes which are not adjusted to the distributed context. A special case of inadequate processes represent inflexible processes (7). Respondents emphasize the problem of rigid processes, where changes are very slowly propagated. Finally, other process barriers reported by the respondents include undefined requirements concerning the tools and infrastructure to be used, resulting in inappropriate tools, missing commitment from the management, above all concerning quality assurance activities related to distributed processes and undefined processes.

Main facets of cultural barriers mentioned by the respondents are not only differences concerning the language (mentioned 16 times). Differences in the awareness of quality (16) or work ethic barriers (8) influence distributed projects, too. One respondent highlights the problem that for some cultural groups it is difficult to express disagreement to customers. Consequently, "nice-to-have" features as well as key features are treated equally, resulting in requirements without priorities. Another participant reports that countries differ in the work ethic with respect to the accuracy of the work as well as to the ability to improvise.

Domain specific barriers mainly subsume differences concerning experience (mentioned 23 times) and the professional formation (18) of distributed teams. Respondents report four kinds of experiences missing in distributed projects: experience in general, experience concerning distributed projects, domain specific experience and experience with specific tools.

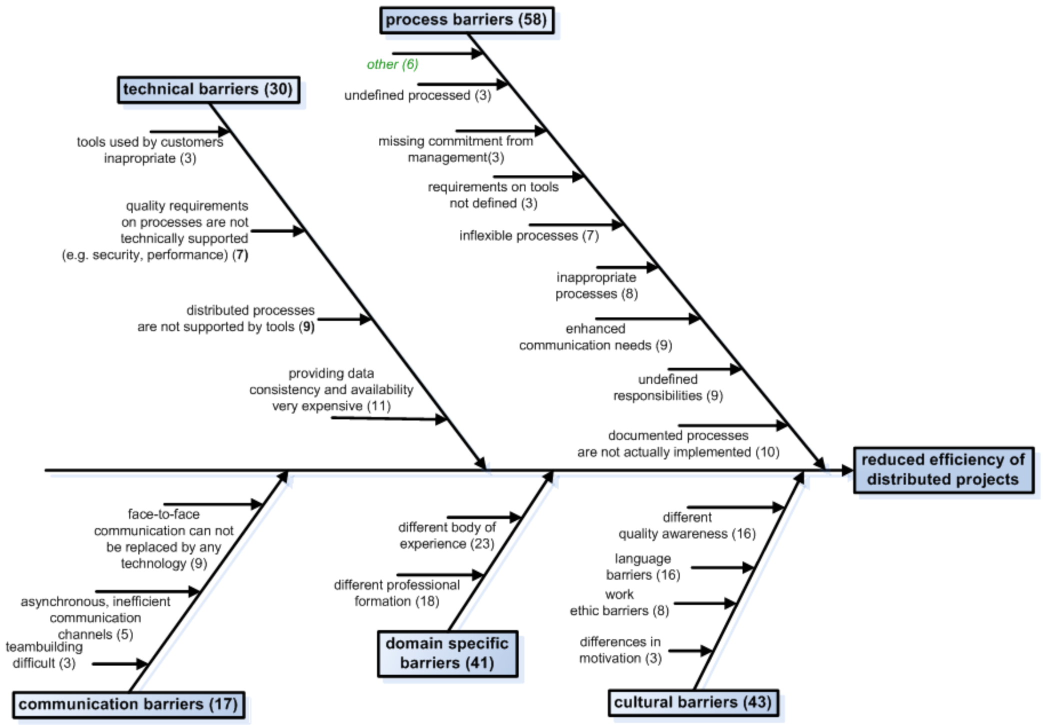
Technical barriers also influence the efficiency of a distributed project. Respondents report that information in form of data is often distributed. Providing consistency and availability of the data are the most important technical barriers (mentioned 11 times).

Copyright © Arbeitsgruppe Software Systems Engineering Heidelberg

Another technical barrier is that tools do not support distributed processes (9) and quality requirements on processes (7). Particularly lacks in security and performance of the tools often prevent their (efficient) usage. Another technical barrier is that tools used by customers are inappropriate and do not integrate well with tools used within the organization (3).

Missing face-to-face communication is a specific communication barrier and it is seen as indispensable even when technological support for synchronous or asynchronous communication is available (mentioned 9 times). One respondent indicates that technology does not replace a convivial evening having a “glass of wine or beer” together. The use of asynchronous, inefficient communication channels represents an often mentioned communication barrier (5). Additionally, respondents also indicate that distributed team building to facilitate communication is a very difficult task (3).

Figure 3. Challenges of distributed development


process barriers (58)
other (6)
undefined processed (3)
missing commitment from management(3)
requirements on tools not defined (3)
inflexible processes (7)
inappropriate processes (8)
enhanced communication needs (9)
undefined responsibilities (9)
documented processes are not actually implemented (10)
technical barriers (30)
tools used by customers inapropriate (3)
quality requirements on processes are not technically supported (e.g. security, performance) (7)
distributed processes are not supported by tools (9)
providing data consistency and availability very expensive (11)
reduced efficiency of distributed projects
face-to-face communication can not be replaced by any technology (9)
asynchronous, inefficient communication channels (5)
teambuilding difficult (3)
communication barriers (17)
different body of experience (23)
different professional formation (18)
domain specific barriers (41)
different quality awareness (16)
language barriers (16)
work ethic barriers (8)
differences in motivation (3)
cultural barriers (43)

4.2. Successful Countermeasures

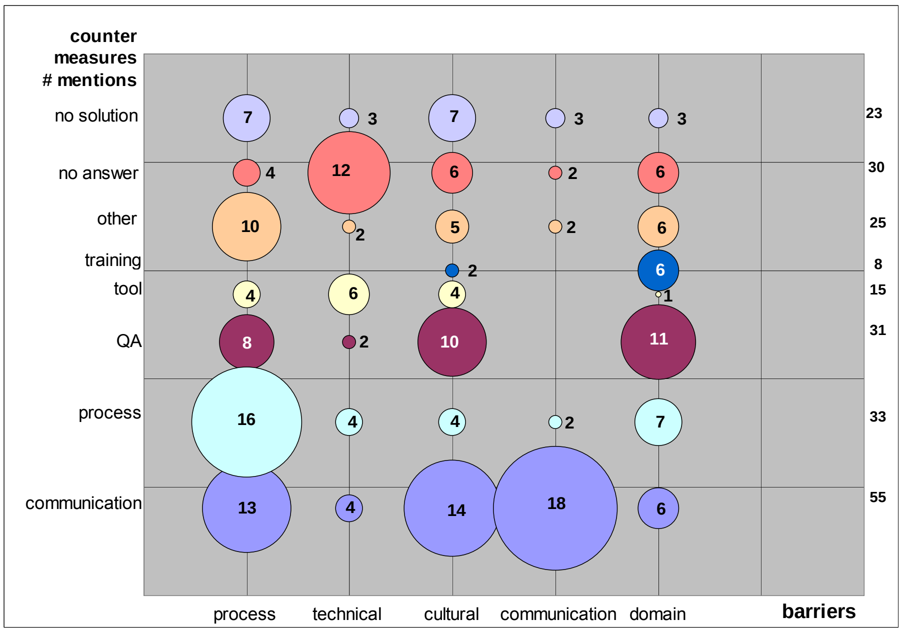
We asked the participants about countermeasures to problems occurring during distributed software engineering, which had successfully been applied. For 136 of the 189 problems, the participants indicated countermeasures. In 30 cases, the participants did not give any answer and in further 23 cases the participants explicitly indicated that there was no successful countermeasure.

The solutions to the barriers presented in Section 4.1 can also be grouped into 5 main categories: communication, process, quality assurance (QA), tool and training. An additional category “other” subsumes solutions which can not be assigned to any of the categories mentioned above. Figure 4 summarizes the responses of the respondents and assigns to each barrier the absolute number of solutions indicated by the participants per

Copyright © Arbeitsgruppe Software Systems Engineering Heidelberg

category. The most important countermeasure to barriers in distributed software development is communication. Above all, intensifying communication is indicated as an important countermeasure to almost all barriers (mentioned 55 times). Above all, communication by email and face-to-face communication were indicated as successful countermeasures in this category.

Figure 4. Countermeasures per barriers – absolute number of indications



Another important group of countermeasures deals with process aspects (mentioned 33 times). Within these mentions, process improvements, a clear definition of responsibilities as well as the definition of process standards were indicated as successful countermeasures in this category. Additionally, the definition of a flexible and iterative development process was also mentioned.

Intensifying quality assurance activities represents another group of countermeasures (mentioned 31 times). Above all, the definition of standards and the performance of more frequent reviews and audits were indicated as successful countermeasures in this category. The definition of standards subsumes the definition of a standard terminology and of a standard language as a countermeasure to communication and domain specific barriers. Additionally, the definition of standard templates has proven of value to overcome domain specific barriers. Finally, intensifying reviewing activities is also a countermeasure to domain specific barriers.

Noticeable is that for about half of all technical problems the participants could not indicate successful solutions.

4.3. Challenges of Distributed Requirements Engineering

In addition to the general problems discussed in the preceding section, in another part of the questionnaire we asked whether there were problems specific to distributed requirements engineering. 58% of the participants answered that they had no problems

Copyright © Arbeitsgruppe Software Systems Engineering Heidelberg

specific to this phase and to distributed software development. 17% answered that there were such problems, but they did not know them, and 25% said there were problems, and these all together listed 122 of such problems.

We wondered how it was possible that 58% reported no specific problems. (As we will show later, in fact the reported problems are specific to distributed requirements engineering, as was the intention of this question.) To find out why this proportion is so high, we first examined to what degree the person's own role influenced his/her answer. Of all those participants who had the role of requirements engineer in the distributed projects, 33% reported detailed problems, 12% said there were such problems, but they did not know which, and 55% said there were none which were specific. So, requirements engineers reported about RE problems only slightly more often than the average participant. Three further explanations for the low percentages of reported specific problems, which probably all are valid to a certain degree, can be: Many of the RE problems observed during distributed software development would have happened likewise in non-distributed projects and therefore were not reported here. Or the participants did not want to answer this question, either because the questionnaire seemed too long to them or because they did not want to give too detailed confidential information about problems.

We also wondered whether many participants did not report RE problems as this phase was not done in a distributed way. 146 participants reported that the RE phase was distributed. For 34% of them, problems were reported, 17% experienced problems without knowing them and 49% seem to have had none. Amazingly, only 57% of those participants who reported RE problems which are specific to distributed software development, also had reported that the RE phase had been performed in a distributed way. These replies are inconsistent. It is possible that the question "Which phase was performed in a distributed way?" was misunderstood by participants, maybe because practitioners do not use the concept of phase. We do not think that the question about problems, which are specific to distributed RE, was misunderstood. Evidence that the participants did understand the question correctly is the fact that such problems which are specific to RE, but not to distributed RE, were rarely reported. Those were found in other studies on RE problems, as in [19]: understanding the users´ needs, conflicts among different customers, how to prioritize requirements, requirements changes.

As was described in Section 2, we asked to name up to three problems (without pre-defined answers) in the part of the questionnaire concerning RE, and in open questions we asked for causes of the problems and for successful countermeasures. In addition to these three problems, further comments concerning RE could be given.

To code the answers to the open questions, we proceeded as follows: We defined the goal of requirements engineering to be the quality of the requirements specification in terms of the quality attributes defined by the IEEE Standard 830-1998 [15] (correctness, unambiguousness, completeness, consistency, verifiability, ranking according to importance and/ or stability, modifiability, traceability) and the efficiency of the specification process. These were the quality criteria we used for the coding of the reported problems.

Each problem reported in the survey was assigned to the quality criterion it endangered. In a second dimension, the reported problems were coded according to the cause of the quality problem observed. These causes were coded according to the types of barriers in Figure 3.

In our MOQARE analysis preceding this survey, we had identified 53 potential misuse cases. Misuse cases combine a cause with a resulting quality problem. (In fact, a misuse case includes much more information, but for our present purpose this

Copyright © Arbeitsgruppe Software Systems Engineering Heidelberg

simplification is useful.) The analysis of the reported requirements engineering problems led to 13 further misuse cases.

Of the 122 problems which the participants reported, 7 could not be coded, because of vague wording. 47 were related to ambiguity of the requirements specification, 44 to the efficiency of the process and 12 to the completeness of the specification. Only five were related to modifiability, three to correctness, two to consistency, one to verifiability, one to prioritization and none to traceability.

The participants named 19 out of the 66 misuse cases more than once. In the following, the ambiguity and efficiency misuse cases are discussed in more detail, because they were clearly named most often. Table 1 illustrates which type of barrier is observed in which context. Significant differences can be observed. For instance, communication barriers play a more important role in RE than in software development in general, and such communication barriers rather lead to inefficient processes than to ambiguous specifications. The ambiguous specification was mostly (at 66%) attributed to domain specific barriers, which were less important for process efficiency, but highly relevant in RE overall. Such domain specific barriers can be lack of technical knowledge as well as domain knowledge. Technical barriers played an even smaller role in RE than in software development in general. Four times, email and phone were mentioned (both together), but because the problem did not spring from the technology itself, we did not count them as technical barriers. Rather these four answers stated that face-to-face communication cannot be replaced by any technology, so we assigned them to communication barriers.

Problem cause	**(a)**	**(b)**	**(c)**	**(d)**
Communication barrier	9%	27%	11%	41%
Domain specific barrier	22%	33%	66%	5%
Cultural b.	23%	16%	21%	18%
Technical b.	16%	3%	0%	5%
Process b.	31%	21%	2%	32%

Tab. 1: Problem causes: (a) in distributed software development in general (data from Figure 3 for comparison); (b) in distributed RE; (c) in distributed RE and leading to ambiguity of the requirements specification or (d) leading to an inefficient RE process (columns add to 100%, i.e. percentage tells the ratio of each barrier within each context)

In addition to the coarse-grained statistics in Table 1, in the following some chosen detail information further illustrates the nature of problems in distributed RE. In the context of ambiguous specifications, half of the cultural barriers were of the type “language barriers”; this is more than in software development in general (compare to Figure 3). In distributed RE, 42% of the domain specific barriers meant different terminology or notation of requirements.

In the context of efficiency of the specification process out of the 18 mentions of communication barriers, 5 stated that face-to-face communication cannot be replaced by indirect respectively distributed communication. The other three sub-types of communication barriers with three answers each were: not enough communication, time zones, and asynchronism of the communication. Among the 14 mentions of process barriers, the most frequent ones were undefined responsibilities (5), high numbers of

Copyright © Arbeitsgruppe Software Systems Engineering Heidelberg

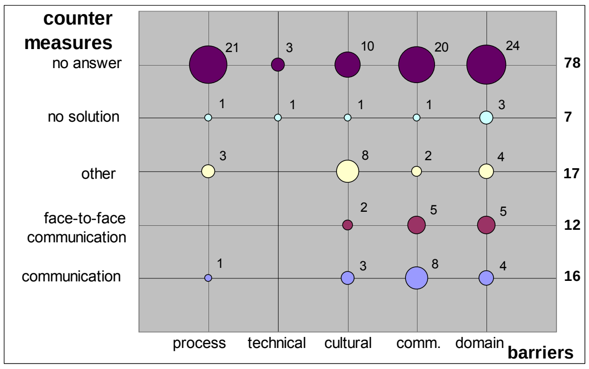
stakeholders as sources of requirements (4), and suitability of processes (3). Out of 8 mentions of cultural barriers, language barriers were mentioned 4 times.

4.4. Successful Countermeasures for Distributed Requirements Engineering

The survey participants were also asked about successfully applied countermeasures for the indicated misuse cases/problems. For only 37 of the 122 problems, such countermeasures were named (as there were often several countermeasures per misuse case, this made 45 countermeasures in all). In seven further cases, the answer explicitly was that there was no successful countermeasure (so far).

As can be seen in Figure 5, the most frequently proposed countermeasures were communication measures (mentioned 16 times) or, more specifically, face-to-face communication (12). This sums up to 28 out of 45 (i.e., 62%). It was proposed to communicate more often, immediately as a question arises, according to formal rules, using a tool (a wiki in this case) and a common terminology. It is remarkable that in distributed development in general (see section 4.1), face-to-face communication was explicitly named only 4 times out of 55 communication measures, i.e. at 7%, and not at 43% as in RE.

Figure 5. Countermeasures for barriers in distributed RE: numbers of mentions ("Comm." = "communication")



The other (17) countermeasures were: Quality Assurance (here: reviews and inspections) reduced ambiguity of the specifications when it is due to language problems, double work done due to unclear responsibilities, and general human communication problems. Training (here: coaching and workshops) helped against culturally caused misunderstandings and the lack of qualification which had led to incomplete requirements (both domain specific barriers). Three times working more was named.

Specific RE countermeasures were proposed for the ambiguity of requirements specifications which is due to different terminology or notation. These were: example requirements, a glossary, early test specifications, standardization of formats, and the definition of minimum standards for documents. When team members differ in working speed, they must be pushed or their work passed on to faster groups. Conflicts among a multitude of stakeholders are handled by the project manager, e.g. by defining goals which are shared by all stakeholders. Other process barriers were tackled by process improvement, i.e. by a formal change process (2) and regular "polling" (1).

Tools were named three times as countermeasures to communication or quality problems. These tools were: video conferencing, VoIP and Wiki. There were two countermeasures which mention email, but one was counted among the communication

Copyright © Arbeitsgruppe Software Systems Engineering Heidelberg

countermeasures (because it said to communicate via email and phone, so the advice was to communicate) and the other among process countermeasures (to send short status notes per email).

5 Summary and Discussion

This technical report presents some of the major results of an online survey among IT professionals who are experienced in distributed software development. In doing so, we focused on the results concerning requirements engineering. The participation rate in this survey was high. So was the average experience of the participants with distributed software development. This shows the practical relevance and representativeness of the topic.

We identified five barriers which influence distributed software engineering projects: process barriers, cultural barriers, domain specific barriers, technical barriers as well as communication barriers. Comparing a former interview study with our study, the authors in [2] also identify communication and domain knowledge issues when developing large software systems (not necessarily in distributed teams). In addition to the problems mentioned in [2], in our study we identified three further problems, which often occur within a distributed project context. The most frequently reported problems concern process barriers. Thus, documented processes are not efficient and appropriate in a distributed context with the result that documented processes are not actually implemented. Another issue mentioned by the participants of our study is related to cultural barriers. This is not surprising, as the study in [2] did not analyse distributed projects. A project team working at one location is more likely to be homogeneous with respect to cultural characteristics than the members of a geographically distributed project. In contrast to the study in [2], the respondents of our study also report technical barriers. Above all, difficulties to provide consistency of distributed data as well as the lack of support for distributed processes are the main issues mentioned by the respondents. In contrast to the study in [2], conflicting requirements are not often mentioned by the respondents. Thus, respondents of our study do not consider this specific to distributed software development.

Altogether, our study shows that problems related to distributed software engineering in general and specific problems related to requirements engineering are similar, but their relative occurrence frequencies vary. For instance, communication barriers are more important in requirements engineering and technical barriers are less important. Moreover, there are particular problems related to requirements engineering. Above all, communication plays a critical role as an important measure against problems.

Our study shows that process-related and human aspects are more important than technical ones. The survey participants did not emphasize tool support, when answering to open questions about problems which they encountered. As communication has been such a frequent countermeasure to many different problems in requirements engineering, we conclude that the main goal of tools and processes must be to support communication.

As can be seen from the literature cited in the introduction, it is not surprising that communication plays such an important role in distributed software development, as a frequent type of barrier as well as a recommended countermeasure to overcome barriers. Thus, this work confirms former, often anecdotic and qualitative findings quantitatively.

Copyright © Arbeitsgruppe Software Systems Engineering Heidelberg

Barrier	[2]	[4]*	[5]	[6]*	[9]*	[14]*	[19]	[20]*	[22]*
Comm.	X	X	X		X	X	X	X	X
Domain specific	X		X			X		X	
Cultural				X				X	
Technical			X						X
Process						X	X	X	X

Tab. 2: Barriers/ problem sources identified by other studies for distributed and non-distributed development (studies which investigate distributed development are marked by *)

Other empirical studies of software development found similar barriers as we did. We summarize their results according to our classification in Table 2. Details of the context of the studies and results are given below:

Curtis et al. [2] found three types of problems in the development of large systems: (1) the thin spread of application domain knowledge, (2) fluctuating and conflicting requirements, (3) and communication bottlenecks and breakdowns. Whether these are more frequent in distributed software engineering than in the "large system development" investigated by Curtis et al. cannot be told, because the authors do not measure their importance quantitatively.

An empirical study of distributed software development found the following types of problems [20]: requirements engineering, lack of standards of the activities in distributed teams, the difficulty to share information and the lack of a well-defined software development process, language barriers and communication, cultural differences, context sharing and trust acquisition among teams. A study of distributed RE [22] found these: communication, planning, management, review process, validation, prototyping, traceability, tool support, knowledge management. [14], in the context of distributed development of embedded systems found: time difference, cultural differences, lack of knowledge of the product. These all are consistent with our findings, although the granularity is not always the same (compare to Figure 3), and the other studies do not quantify the importance of the problems.

One interview study of distributed software development identified some further problems, not found in our survey [14]. These are: openness of communication between partners, problem hiding in customer-supplier relations, unclear assignments, trust, agreeing on intellectual property rights, reliability of the partners´ development schedule, continuation of the collaboration in the future, predicting the most sales-boosting features, quality of the product, becoming too dependent on one partner, competence of new partners, weakening of one's own competence. One can wonder whether such problems are rather mentioned in an interview study than in an online survey because of higher trust and openness towards the interviewer, or whether they were not considered to be specific to distributed software development.

As was mentioned in section 4.3, such problems which are specific to RE, but not to distributed RE, were rarely reported during our survey, such as: understanding the users´ needs, conflicts among different customers, how to prioritize requirements, requirements changes [19], or: package considerations (analysis of COTS products), level of detail of process models, examining current system, user participation, managing uncertainty,

Copyright © Arbeitsgruppe Software Systems Engineering Heidelberg

CASE RE tools, project management [5]. This is because we asked for problems which are specific for distributed RE, and also shows that the participants focused on these.

Countermeasure	[2]	[4]*	[5]	[9]*	[14]*	[20]*	[22]*
Communication			X	X		X	
Face-to-face comm.		X			X	X	
Training	X		X			X	
Tool	X			X			X
Process			X			X	
Others					X		

Tab. 3: Countermeasures identified by other studies for distributed and non-distributed development (studies which investigate distributed development are marked by *)

Table 3 presents an overview of countermeasures proposed by other studies. The following were proposed by practitioners: to increase application domain knowledge, tools and methods must allow change, appropriate communication media [2]; planning, training, standardization, requirements engineering (face to face if possible), trust and integration [20].

Although the other studies did not measure the importance of these countermeasures quantitatively, it seems that tools and communication media as well as training had a lower weight in our study. We believe that this is because these countermeasures are not successful or not perceived as being so by the team members (we explicitly asked for successful countermeasures to the named problems). For instance, in [14] it was found that most tools do not support collaborative development well enough. Although practitioners did not report tools as a major problem neither in [14] nor in our study, this can serve to explain why tools are rarely seen as solution of problems in distributed software development.

From literature, one can edit lists of countermeasures which are more comprehensive than those found in empirical studies [14]. Many of these countermeasures were not mentioned by the practitioners in our study and in those studies cited above. There can be several explanations for this observation. Either these countermeasures are not known to practitioners, are not applied or are not perceived as being useful. Such countermeasures were: synchronisation of main milestones, clear decision-making practices, decoupling the work across different sites, one project leader, relationship management, architectural practices, frequent deliveries, frequent and incremental integration, up-to-date documentation [14].

Possible threats to the validity of our results include for instance, that a high proportion of the participants of this survey were developers and designers. This does not necessarily mean that this adds a significant bias to the results as most of the projects were small; so also the developers probably had an overview of the project, and the questionnaire always offered the option to answer “I do not know”, e.g. concerning requirements engineering problems. However, our analyses in the beginning of section 4.3 show that requirements engineering problems evidently have also been known to

Copyright © Arbeitsgruppe Software Systems Engineering Heidelberg

participants other than the requirements engineer. The answers to our questions were subjective, i.e. the participants named the problems which were most memorable to them and the countermeasures which they believed were most efficient. The relevance of these problems and the efficiency of countermeasures have not been shown by statistical analyses of project data, which was not our purpose. Some practices which usually are advised in technical literature, like using a glossary in requirements engineering, rarely appeared in the answers. This does not necessarily mean that they are not used in practice, but this shows that their lack has not been a major problem (either because a glossary is less important or used without saying), and that such a practice was not the most important solution to a problem. As we have discussed before, some measures are necessary, but not sufficient preconditions of good work (like the tools) and therefore are presumably not mentioned in this survey. We did not compare our results quantitatively with such for non-distributed projects so far, mainly because our focus was to investigate the state of the practice. Some of the identified problems also occur in non-distributed projects. However, the comments and examples given by the participants indicate that they quite well understood that we asked for problems which were specific to distributed work. The above must be kept in mind when interpreting the survey results. Nevertheless, we think that our results are a good basis for investigating project problems and practices as perceived by the team members, because of the high number of participants and the amount of data.

Future work will focus on further analyses, especially on software development phases other than the requirements engineering. Furthermore, we expect that the in-depth analysis of correlations will lead to further interesting insights, e.g. whether some problems are more frequent in big projects than in small ones.

Acknowledgements

We would like to thank the numerous participants of the survey for their time, trust and meaningful answers, and Barbara Paech for her constructive suggestions, comments and references. We also appreciate the financial and organizational support of the survey by the MFG, especially by Eike Bieber. We also would like to thank Heise for publishing the survey address in their news ticker, which significantly contributed to the high participation. This work was funded by the research network PRIMIUM.

6 References

[1] C. Cook and N. Churcher, “An Extensible Framework for Collaborative Software Engineering”, *Proceedings of the 10th Asia-Pacific Software Engineering Conference APSEC’03*, IEEE, 2003, pp. 290–301.

[2] B. Curtis, H. Krasner, and N. Iscoe, “A field study of the software design process for large systems”, *Communications of the ACM 31 (11)*, 1988, pp.1268-1287.

[3] D.L. Dean, J.D. Lee, M.O. Pendergast, A.M. Hickey, J.F. Nunamaker, “Enabling the Effective Involvement of Multiple Users: Methods and Tools for Collaborative Software Engineering”, *Journal of Management Information Systems 14 (3),* 1998, pp. 179–222.

[4] A.H. Dutoit, J. Johnstone, and B. Bruegge, “Knowledge scouts: Reducing communication barriers in a distributed software development project”, *Proceedings of the Eighth Asia-Pacific on Software Engineering Conference APSEC,* IEEE, Dec. 2001, pp. 427-430.

Copyright © Arbeitsgruppe Software Systems Engineering Heidelberg

[5] K. El Emam and N.H. Madhavji, “A field study of requirements engineering practices in information systems development”, *Proceedings of the International Symposium on Requirements Engineering,* IEEE, 1995, pp.68-80.

[6] J. Grieb and U. Lindemann: “Design Communication in Industry: A Survey Analysis”, *International Conference on Engineering Design ICED 05,* Melbourne, Aug. 15 – 18, 2005, pp. 586-587.

[7] R.E. Grinter, J.D. Herbsleb, and D. E. Perry, “The geography of coordination: Dealing with distance in R&D work”, *Proceedings of the ACM Conference on Supporting Group Work (GROUP 99)*, Phoenix, AZ, November 14-17, pp. 306-315.

[8] J. Grudin, “Why CSCW applications fail: problems in the design and evaluation of organization of organizational interfaces “, *Proceedings of the 1988 ACM Conference on Computer-Supported Cooperative Work,* ACM, New York, 1988, pp.85-93.

[9] C. Gutwin, R. Penner, and Kevin Schneider, “Group Awareness in Distributed Software Development”, *Proceedings of the 2004 ACM conference on Computer supported cooperative work CSCW’04*, IEEE, Chicago, Illinois, USA, November 6–10, 2004, pp.72-81.

[10] A. Herrmann, D. Kerkow, and J. Doerr, “Exploring the Characteristics of NFR Methods – a Dialogue about two Approaches”, *REFSQ - Workshop on Requirements Engineering for Software Quality (2007), Foundations of Software Quality*, to be published.

[11] A. Herrmann, B. Paech, “Quality Misuse”, *REFSQ - Workshop on Requirements Engineering for Software Quality (2005), Foundations of Software Quality, Essener Informatik Beiträge,* Universität Duisburg-Essen, Essen, 2005, pp. 193-199.

[12] T. Hildenbrand, F. Rothlauf and A. Heinzl, “Ansätze zur kollaborativen Softwareerstellung”, WIRTSCHAFTS-INFORMATIK 49 (Special Issue), 2007, pp. S72–S80.

[13] I. Hoh and R. Siddharta Roy, “Visualization issues for software requirements negotiation”, *Proceedings of the 25th Annual International Computer Software and Applications Conference COMPSAC,* 8-12 Oct 2001, pp.10-15.

[14] J. Hyysalo, P. Parviainen, and M. Tihinen, “Collaborative Embedded Systems Development: Survey of State of the Practice”, Proceedings of the 13th Annual IEEE International Symposium and Workshop on Engineering of Computer Based Systems (ECBS06), IEEE, 2006.

[15] IEEE, *Std. 830-1998: IEEE Recommended Practice for Software Requirements Specification*, IEEE, Washington, 1998

[16] MFG Baden-Württemberg, http://www.english.doit-online.de/cms/About+us/ MFG+Baden-W%FCrttemberg, last visited May 2007

[17] B.A. Kitchenham and S.L. Pfleeger "Principles of survey research: part 3: constructing a survey instrument", *SIGSOFT Softw. Eng. Notes* vol. 27, no. 2, pp. 20-24, 2002.

[18] B. Kitchenham and S.L. Pfleeger "Principles of survey research part 6: data analysis", *SIGSOFT Softw. Eng. Notes* vol. 28, no. 2, 2003, pp. 24-27.

[19] M. Lubars, C. Potts, and Ch. Richter, “A review of the state of the practice in requirements modeling”, *Proceedings of the International Symposium on Requirements Engineering,* IEEE, 1992, pp.2-14.

[20] R. Prikladnicki, J.L.N. Audy, and R. Evaristo, “An

[21] Empirical Study on Global Software Development: Offshore Insourcing of IT Projects”, *Proceedings of the International Workshop on Global Software Development,* International Conference on Software Engineering (ICSE 2004), IEEE, Edinburgh, Scotland, May 24, 2004, pp. 53-58.

[22] B. Regnell, M. Höst, J. Natt och Dag, P. Beremark, and T. Hjelm, ”An Industrial Case Study on Distributed Prioritisation in Market-Driven Requirements Engineering for Packaged Software, *Requirements Engineering Journal 6(1)*, 2001, pp. 51–62.

Copyright © Arbeitsgruppe Software Systems Engineering Heidelberg

[23] D. Zowghi, D. Damian, and R. Offen, „Field Studies of Requirements Engineering in a Multi-Site Software Development Organization“, *Proceedings of the Australian Workshop on Requirements Engineering,* Sydney, Nov. 2001.

[24] T. Illes-Seifert, A. Herrmann, M. Geisser, T. Hildenbrand, "The Challenges of Distributed Software Engineering and Requirements Engineering: Results of an Online Survey", *GREW07: 1st International Global Requirements Engineering Workshop*, 27th august 2007, Munich, Germany

[25] Geisser M, Herrmann A, Hildenbrand T, Illes-Seifert T, „Verteilte Softwareentwicklung und Requirements Engineering: Ergebnisse einer Online-Umfrage, *Objektspektrum*, 2007, to appear.

Copyright © Arbeitsgruppe Software Systems Engineering Heidelberg

Annexes

Annex A: The original Questionnaire in German

Umfrage zur verteilten Software-Entwicklung

Der Trend zu Unterbeauftragung, Outsourcing, Offshoring, aber auch niederlassungs- und länderübergreifende Zusammenarbeit innerhalb einer großen Firma oder Behörde führt dazu, dass über mehrere Standorte verteilt vorhandenes Wissen und Ressourcen genutzt werden können. Diese Zusammenarbeit muss jedoch auch technisch unterstützt werden. Das Ziel dieser Umfrage ist es, den Stand der Praxis in Bezug auf verteilte Software-Entwicklung zu erheben.

Eine **verteilte Software-Entwicklung** in dieser Umfrage hat folgende Charakteristika:

– alle/einige der Beteiligten verwenden überwiegend **verteilte Techniken** um miteinander zu kommunizieren (z.B. weil auf Grund der geographischen Entfernung nicht anders möglich)
– Kommunikation findet statt um zu diskutieren, **Entscheidungen** zu treffen, **Informationen** und **Wissen** auszutauschen und zu konsolidieren sowie gemeinsame Arbeiten der Softwareentwicklung auszuführen.

  Beispielsweise kann die Diskussion durch E-Mail, Wiki, Forum, Chat, Netmeeting, IP-Telefonie unterstützt werden.
  Informations- und Wissensaustausch kann zum Beispiel durch Repositories zur gemeinsamen Datenablage und -Verwaltung aber auch durch Werkzeuge zur Unterstützung unterschiedlicher Software-Engineering Aktivitäten unterstützt werden.

Wir würden uns sehr freuen, wenn Sie mit verteilter Software-Entwicklung Erfahrung haben und an unserer Umfrage teilnehmen. Das Ausfüllen des Online-Fragebogens dauert ca. 20 Minuten. Wenn Sie uns Ihre E-Mail-Adresse mitteilen, senden wir Ihnen gerne die Ergebnisse der Umfrage zu.

Mit freundlicher Unterstützung der MFG Baden-Württemberg mbH verlosen wir unter den Teilnehmern der Umfrage einen Gutschein über die Teilnahme an einem Tagesseminar im Rahmen der MFG Akademie im Wert von 238,- EUR.

Diese Umfrage wird im Rahmen einer Kooperation zwischen dem Lehrstuhl für Software Engineering an der Universität Heidelberg (www-swe.informatik.uni-heidelberg.de) und dem Lehrstuhl für ABWL und Wirtschaftsinformatik an der Universität Mannheim (wifo1.bwl.uni-mannheim.de) durchgeführt.

Bei Fragen können Sie sich wenden an:

Copyright © Arbeitsgruppe Software Systems Engineering Heidelberg

Dr. Andrea Herrmann
herrmann@informatik.uni-heidelberg.de
Tel.: 06221-545816

Fragen zu verteilten Software-Projekten

Seite 2:

In wie vielen Projekte mit verteilter Software-Entwicklung haben Sie bereits gearbeitet? (Anzahl) _______

Mittlerer Projektumfang in Personenmonaten: _______

Größe der Firma/ Organisation (oder Firmen/ Organisationen), in denen Sie an verteilten Software-Projekte teilgenommen haben
(Mehrfachantworten möglich)
☐ > 10.000 Mitarbeiter/innen
☐ 1000 – 10.000
☐ 100 - 999
☐ < 100

Welche Rolle(n) haben Sie in diesen Projekten wahrgenommen?
(Mehrfachantworten möglich)
☐ Projektleiter/in
☐ Anforderungsingenieur/in
☐ Software-Architekt/in
☐ Entwickler/in
☐ Tester/in
☐ Andere, nämlich: ______________________________

Welche Phasen des Software-Entwicklungsprozesses wurden verteilt durchgeführt?
(Mehrfachantworten möglich)
☐ Anforderungsanalyse
☐ Architektur/Design
☐ Implementierung
☐ Test
☐ Betrieb und Wartung
☐ Sonstige, und zwar: ____________________________

Von wem wurde die verteilte Technologie genutzt?
(Mehrfachantworten möglich)
☐ Projektleiter
☐ Vertrieb
☐ Einkauf
☐ Produktmanager
☐ Sonstige Manager
☐ Anforderungsingenieur (Requirements Engineer)
☐ Sonstiger Berater
☐ Software-Architekt

Copyright © Arbeitsgruppe Software Systems Engineering Heidelberg

☐ Entwickler
☐ Qualitätssicherer
☐ Wartung und Support
☐ Hardware-Betreiber oder –Lieferant
☐ (zukünftige) Benutzer
☐ Projektlenkungsausschuss
☐ Change Control Board
☐ Sonstige, nämlich: ____

Seite 5:

Welche verteilte Technologie wurde verwendet?
(Mehrfachantworten möglich)
zur Unterstützung der Kommunikation während der verteilten Software Entwicklung
☐ Telefon
☐ Telefonkonferenz
☐ VoIP, z.B. Skype
☐ E-Mail
☐ Online-Forum
☐ Wiki
☐ Chat
☐ Internetbasierte Videokonferenzen
☐ Anderes, nämlich: ____________________________

zur Unterstützung des Informationsaustausches während der verteilten Software Entwicklung
☐ Zentrales Repository für Dokumentenablage und -management
☐ Versionierungssystem für Code, z.B. CVS oder ClearCase
☐ Software-Engineering Werkzeug, z.B. Anforderungsmanagement-Werkzeug, Test-Werkzeug
☐ (Projekt-)Management-Werkzeug
☐ Anderes, nämlich: ____________________________

Vertreter wie vieler Firmen oder Organisationseinheiten kommunizierten in diesen Projekten über die verteilte Technologie miteinander? _______
☐ weiß ich nicht

Wie viele Personen kommunizierten durchschnittlich pro Projekt über die verteilte Technologie miteinander? _______
☐ weiß ich nicht

Wie viele Personen arbeiteten durchschnittlich in diesen Projekten? _______
☐ weiß ich nicht

Mit welcher Art von verteilter Software-Entwicklung haben Sie Erfahrung?
(Mehrfachantworten möglich)
☐ Offshoring in Länder auf anderen Kontinenten wie Indien oder China
☐ Unterbeauftragung (einschließlich Outsourcing).
☐ Gleichberechtigte Zusammenarbeit mehrerer Firmen

Copyright © Arbeitsgruppe Software Systems Engineering Heidelberg

☐ Wir haben innerhalb unserer Organisation national, aber niederlassungsübergreifend zusammengearbeitet.
☐ Wir haben innerhalb unserer Organisation international zusammengearbeitet, wobei die Beteiligten in verschiedenen Ländern waren.
☐ Wir haben innerhalb einer Niederlassung derselben Organisation zusammengearbeitet, aber trotzdem verteilte Kommunikationsmittel verwendet, um uns abzustimmen.
☐ Sonstiges, nämlich: ____
☐ Weiß nicht

____

Zusammen mit Kunden welcher Branche wurde die Software-Entwicklung verteilt durchgeführt?
(Mehrfachantworten möglich)
☐ Kommerzieller Bereich – Bank, Versicherung
☐ Technischer Bereich – Maschinenbau, Chemie, Elektrotechnik, Telekommunikation, Transport
☐ Öffentlicher Bereich – Verwaltung, Regierung
☐ Dienstleistung – Ausbildung, Beratung, IT-Dienstleistung
☐ Software
☐ Sonstiges, nämlich: ______________________________

Kommentare:
____
____
____
____

Seite 8:
Schwierigkeiten und Lösungen

Im Folgenden interessieren uns Schwierigkeiten, die speziell bei der verteilten Software-Entwicklung auftreten und hier häufiger sind oder schädlichere Folgen haben als bei der „traditionellen“, nicht-verteilten Software-Entwicklung.

1. *Schwierigkeiten bei der Prozessunterstützung und -dokumentation*
2. *Requirements Engineering (Erhebung, Dokumentation, Analyse und Prüfung der Anforderungen)*
3. *Architekturentwurf und Implementierung*
4. *Testen*

Seite 9:
1. Schwierigkeiten bei der verteilten Prozessunterstützung und -dokumentation

Copyright © Arbeitsgruppe Software Systems Engineering Heidelberg

*Welche der folgenden **Schwierigkeiten** traten in den Projekten mit verteilter Technologie wie **oft** auf, mit welchem **Schaden** und wegen welcher **Ursache**? Welche **Gegenmaßnahmen** (technische, organisatorische,...) haben Sie erfolgreich eingesetzt?*

Projektdokumentation war nicht eindeutig

Nie	selten immer	oft	meist
<5 % >95%	5-25%	25-75	75-95
☐	☐ ☐	☐	☐

verursachter Schaden: ☐ gering ☐ mittel ☐ hoch

Ursachen und erfolgreiche Gegenmaßnahmen:

Projektdokumentation war unvollständig dokumentiert

Nie	selten immer	oft	meist
<5 % >95%	5-25%	25-75	75-95
☐	☐ ☐	☐	☐

verursachter Schaden: ☐ gering ☐ mittel ☐ hoch

Ursachen und erfolgreiche Gegenmaßnahmen:

Seite 10:

Projektdokumentation war oder blieb widersprüchlich

Nie	selten immer	oft	meist

Copyright © Arbeitsgruppe Software Systems Engineering Heidelberg

<5 %	5-25%	25-75	75-95	>95%
☐	☐	☐	☐	☐

verursachter Schaden: ☐ gering ☐ mittel ☐ hoch

Ursachen und erfolgreiche Gegenmaßnahmen:

Projektdokumentation war falsch

Nie	selten	oft	meist	immer
<5 %	5-25%	25-75	75-95	>95%
☐	☐	☐	☐	☐

verursachter Schaden: ☐ gering ☐ mittel ☐ hoch

Ursachen und erfolgreiche Gegenmaßnahmen:

Seite 11:

Projektdokumentation konnte nicht oder nicht sinnvoll/ nachvollziehbar an Änderungen angepasst werden

Nie	selten	oft	meist	immer
<5 %	5-25%	25-75	75-95	>95%
☐	☐	☐	☐	☐

verursachter Schaden: ☐ gering ☐ mittel ☐ hoch

Ursachen und erfolgreiche Gegenmaßnahmen:

Der Prozess für die Projektdokumentation war nicht effizient

Nie	selten	oft	meist	immer
<5 %	5-25%	25-75	75-95	>95%
□	□	□	□	□

verursachter Schaden: □ gering □ mittel □ hoch

Ursachen und erfolgreiche Gegenmaßnahmen:

Das Werkzeug für die verteilte Software-Entwicklung bot keine effiziente Prozessunterstützung

Nie	selten	oft	meist	immer
<5 %	5-25%	25-75	75-95	>95%
□	□	□	□	□

verursachter Schaden: □ gering □ mittel □ hoch

Ursachen und erfolgreiche Gegenmaßnahmen:

Copyright © Arbeitsgruppe Software Systems Engineering Heidelberg

Es gab keine oder keine qualitativ hochwertige Prozessdokumentation

Nie selten oft meist
immer
<5 % 5-25% 25-75 75-95
>95%
☐ ☐ ☐ ☐
☐

verursachter Schaden: ☐ gering ☐ mittel ☐ hoch

Ursachen und erfolgreiche Gegenmaßnahmen:

Das Werkzeug unterstützte keine effiziente Wiederverwendung von Inhalten

Nie selten oft meist
immer
<5 % 5-25% 25-75 75-95
>95%
☐ ☐ ☐ ☐
☐

verursachter Schaden: ☐ gering ☐ mittel ☐ hoch

Ursachen und erfolgreiche Gegenmaßnahmen:

Kommentare zur verteilten Prozessunterstützung und -dokumentation:

Seite 14:

2. Requirements Engineering/ Anforderungserhebung und -analyse

Welche Schwierigkeiten traten in dieser Phase bei der verteilten Software-Entwicklung auf?

Copyright © Arbeitsgruppe Software Systems Engineering Heidelberg

□ Es gab **keine besonderen Schwierigkeiten** in diesem Bereich, die auf die verteilte Software-Entwicklung zurückzuführen sind.
□ Es gab Schwierigkeiten in diesem Bereich, die auf die verteilte Software-Entwicklung zurückzuführen sind, **ich kenne sie aber nicht**.
□ Es gab folgenden Schwierigkeiten

Seite 15:

1.)

Nie	selten	oft	meist	immer
<5 %	5-25%	25-75	75-95	>95%
□	□	□	□	□

verursachter Schaden: □ gering □ mittel □ hoch

Ursachen und erfolgreiche Gegenmaßnahmen:

2.)

Nie	selten	oft	meist	immer
<5 %	5-25%	25-75	75-95	>95%
□	□	□	□	□

verursachter Schaden: □ gering □ mittel □ hoch

Ursachen und erfolgreiche Gegenmaßnahmen:

3.)

Nie	selten	oft	meist	immer
<5 %	5-25%	25-75	75-95	>95%

Copyright © Arbeitsgruppe Software Systems Engineering Heidelberg

☐	☐	☐	☐	☐

verursachter Schaden: ☐ gering ☐ mittel ☐ hoch

Ursachen und erfolgreiche Gegenmaßnahmen:

Kommentare zum verteilten Requirements Engineering:

Seite 16:

3. Architekturentwurf und Implementierung

Welche Schwierigkeiten traten in dieser Phase bei der verteilten Software-Entwicklung auf?

☐ Es gab **keine besonderen Schwierigkeiten** in diesem Bereich, die auf die verteilte Software-Entwicklung zurückzuführen sind.

☐ Es gab Schwierigkeiten in diesem Bereich, die auf die verteilte Software-Entwicklung zurückzuführen sind, **ich kenne sie aber nicht**.

☐ Es gab folgenden Schwierigkeiten:

Seite 17:

1.)____

Nie	selten	oft	meist	immer
<5 %	5-25%	25-75	75-95	>95%
☐	☐	☐	☐	☐

verursachter Schaden: ☐ gering ☐ mittel ☐ hoch

Ursachen und erfolgreiche Gegenmaßnahmen:

Copyright © Arbeitsgruppe Software Systems Engineering Heidelberg

2.)____

Nie	selten	oft	meist	immer
<5 %	5-25%	25-75	75-95	>95%
☐	☐	☐	☐	☐

verursachter Schaden: ☐ gering ☐ mittel ☐ hoch

Ursachen und erfolgreiche Gegenmaßnahmen:

3.)____

Nie	selten	oft	meist	immer
<5 %	5-25%	25-75	75-95	>95%
☐	☐	☐	☐	☐

verursachter Schaden: ☐ gering ☐ mittel ☐ hoch

Ursachen und erfolgreiche Gegenmaßnahmen:

Kommentare zu verteiltem Architekturentwurf und Implementierung:

Seite 18:

4. Testen

Copyright © Arbeitsgruppe Software Systems Engineering Heidelberg

Welche Schwierigkeiten traten in dieser Phase bei der verteilten Software-Entwicklung auf?
□ Es gab **keine besonderen Schwierigkeiten** in diesem Bereich, die auf die verteilte Software-Entwicklung zurückzuführen sind.
□ Es gab Schwierigkeiten in diesem Bereich, die auf die verteilte Software-Entwicklung zurückzuführen sind, **ich kenne sie aber nicht**.
□ Es gab folgenden Schwierigkeiten:

Seite 19:

1.)

Nie	selten	oft	meist	immer
<5 %	5-25%	25-75	75-95	>95%
□	□	□	□	□

verursachter Schaden: □ gering □ mittel □ hoch

Ursachen und erfolgreiche Gegenmaßnahmen:

2.)

Nie	selten	oft	meist	immer
<5 %	5-25%	25-75	75-95	>95%
□	□	□	□	□

verursachter Schaden: □ gering □ mittel □ hoch

Ursachen und erfolgreiche Gegenmaßnahmen:

3.)

Copyright © Arbeitsgruppe Software Systems Engineering Heidelberg

Nie	selten	oft	meist	immer
<5 %	5-25%	25-75	75-95	>95%
☐	☐	☐	☐	☐

verursachter Schaden: ☐ gering ☐ mittel ☐ hoch

Ursachen und erfolgreiche Gegenmaßnahmen:

Kommentare zum verteilten Testen:

Seite 20:

☐ **Ja, ich möchte die Auswertung der Umfrage per E-Mail zugesandt bekommen.**
Meine E-Mail-Adresse: ____
(Ihre E-Mail-Adresse wird nur zu diesem Zweck verwendet und nach Versand der Umfrageergebnisse gelöscht.)

Vielen Dank für die Beantwortung des Fragebogens!

Copyright © Arbeitsgruppe Software Systems Engineering Heidelberg

Annex B: The English Translation of the Questionnaire

Survey about distributed Software Development

The trend towards sub-contracting, outsourcing, and off-shoring, as well as the collaboration with partner organizations or within an organization at different locations (nationally and internationally) requires the use of knowledge and resources distributed over multiple locations. However, this collaboration must also be supported technically. It is the objective of this survey to investigate the state of the practice in distributed software development.

In this survey, we define **distributed software development** by the following characteristics:

- All or at least some participants of a software project predominantly use **distributed technologies** for team communication (e.g. because this is not possible otherwise due to geographical distance)
- Communication takes place for discussing, **decision**-making, **information** and **knowledge** exchange and consolidation, as well as joint work on software development.

The communication can, for instance, be supported by email, wiki, online forum, chat, netmeeting, voice-over-IP. Information and knowledge exchange can be supported by document management systems, version control systems but also by tools for different distributed software engineering activities.

We would be glad if you have experience with distributed software development and participated in our survey. Completing the online questionnaire takes about 20 minutes. If you give us your email address, we will be pleased to send you the survey results.

With friendly sponsorship by the MFG Baden-Württemberg mbH, one of the survey participants can win a voucher for day seminar at the MFG Akademie of the value of 238,- EUR.

This survey is realized by a co-operation of the Chair for Software Engineering at the University of Heidelberg (www-swe.informatik.uni-heidelberg.de) and the Chair of Applied Business Adminstration and Business Informatics of the University of Mannheim (wifo1.bwl.uni-mannheim.de).

In case of questions, you can contact:
Dr. Andrea Herrmann
herrmann@informatik.uni-heidelberg.de
Tel.: 06221-545816

Copyright © Arbeitsgruppe Software Systems Engineering Heidelberg

Questions about distributed software projects

Page 2:

In how many projects with distribtued software development have you worked so far?
(Number) _______

Average project size in person months: _______

Size of the company/ organization (or companies/ organizations), in which you have participated in distributed software projects
(multiple answers possible)
☐ > 10.000 employees
☐ 1000 – 10.000
☐ 100 - 999
☐ < 100

Which role(s) did you have in these projects?
(multiple answers possible)
☐ project manager
☐ requirements engineer
☐ software architect
☐ developer
☐ tester
☐ others, namely: ______________________________

Which project phases of the software development process have been done in a distributed way?
(multiple answers possible)
☐ requirements analysis
☐ architectural design
☐ implementation
☐ testing
☐ operation and maintenance
☐ others, namely ____________________________

Who did use the distributed technology?
(multiple answers possible)
☐ project manager
☐ sales
☐ purchase
☐ product manager
☐ other managers
☐ requirements engineer
☐ other consultant
☐ software architect
☐ developer

Copyright © Arbeitsgruppe Software Systems Engineering Heidelberg

☐ quality manager
☐ maintenance and support
☐ hardware operator or supplier
☐ (future) user
☐ project steering committee
☐ change control board
☐ others, namely: ____

Page 5:

Which distributed technology was used?
(multiple answers possible)
for supporting the communication during distributed software development
☐ telephone
☐ conferencing call
☐ VoIP, e.g. Skype
☐ email
☐ online forum
☐ wiki
☐ chat
☐ videoconference via internet
☐ others, namely: ____________________________

for supporting the information exchange during distributed software development
☐ central repository for document management
☐ version control system for code, e.g. CVS or ClearCase
☐ software engineering tools, e.g. requirements management tool, test tool
☐ (project) management tool
☐ others, namely: ____________________________

Persons from how many companies or organizational units did communicate in these projects via distributed technology? _______
☐ I don´t know.

How many persons (in average) did communicate in these projects via distributed technology? _______
☐ I don´t know.

How many persons worked in these projects in average? _______
☐ I don´t know.

With which type of distributed software development do you have experience?
(multiple answers possible)
☐ offshoring in countries on other continents like India or China
☐ sub-contracting (including outsourcing)
☐ collaboration with partner organizations
☐ We have collaborated within our organization nationally, but at different locations.
☐ We have collaborated within our organization internationally, the participants being in different countries.

Copyright © Arbeitsgruppe Software Systems Engineering Heidelberg

☐ We have collaborated within one location of the same organization, but nevertheless used distributed technology for coordinating.
☐ Others, namely: ____
☐ I don´t know.

____

The distributed software development was done together with customers from which business domain?
(multiple answers possible)
☐ commercial sector – banking, insurance
☐ technical sector – mechanical engineering, chemistry, electrical engineering, telecommunication, and transport
☐ public sector – administration, government
☐ service – education, consulting, IT services
☐ software
☐ Others, namely: _______________________________

Comments:
____
____
____
____

Page 8:
Challenges and Solutions

In the following, we are interested in challenges, which emerge specifically during distributed software development and which here are more frequent or have more harmful consequences than during "traditional", non-distributed software development.

1. *Challenges during process support and process documentation*
2. *Requirements engineering (elicitation, documentation, analysis and validation of the requirements)*
3. *Architectural design and implementation*
4. *Testen*

Page 9:
1. Challenges of the distributed process support and process documentation

Which of the following ***challenges*** *did emerge* ***how often****, with which* ***damage*** *and due to which* ***cause****? Which* ***countermeaseures*** *(technical, organizational, …) have you applied successfully?*

Project documentation was ambiguous

never rarely often mostly always

<5 % 5-25% 25-75 75-95
>95%
□ □ □ □
□

caused damage: □ low □ medium □ high

Causes and successful countermeasures:

Project documentation was incomplete

never rarely often mostly
always
<5 % 5-25% 25-75 75-95
>95%
□ □ □ □
□

caused damage: □ low □ medium □ high

Causes and successful countermeasures:

Page 10:

Project documentation was or stayed contradictory

never rarely often mostly
always
<5 % 5-25% 25-75 75-95
>95%
□ □ □ □
□

caused damage: □ low □ medium □ high

Causes and successful countermeasures:

Copyright © Arbeitsgruppe Software Systems Engineering Heidelberg

Project documentation was wrong

never	rarely	often	mostly	always
<5 %	5-25%	25-75	75-95	>95%
☐	☐	☐	☐	☐

caused damage: ☐ low ☐ medium ☐ high

Causes and successful countermeasures:

Page 11:

Project documentation could not be adapted to changes, or not reasonable/ traceably

never	rarely	often	mostly	always
<5 %	5-25%	25-75	75-95	>95%
☐	☐	☐	☐	☐

caused damage: ☐ low ☐ medium ☐ high

Causes and successful countermeasures:

Copyright © Arbeitsgruppe Software Systems Engineering Heidelberg

The project documentation process was inefficient

never	rarely	often	mostly
	always		
<5 %	5-25%	25-75	75-95
>95%			
☐	☐	☐	☐
	☐		

caused damage: ☐ low ☐ medium ☐ high

Causes and successful countermeasures:

The tool for distributed software development did not offer efficient process support.

never	rarely	often	mostly
	always		
<5 %	5-25%	25-75	75-95
>95%			
☐	☐	☐	☐
	☐		

caused damage: ☐ low ☐ medium ☐ high

Causes and successful countermeasures:

There was no or no high-quality process documentation

never	rarely	often	mostly
	always		
<5 %	5-25%	25-75	75-95
>95%			
☐	☐	☐	☐
	☐		

Copyright © Arbeitsgruppe Software Systems Engineering Heidelberg

caused damage: ☐ low ☐ medium ☐ high

Causes and successful countermeasures:

The tool did not support an efficient reuse of content

never	rarely	often	mostly	always
<5 %	5-25%	25-75	75-95	>95%
☐	☐	☐	☐	☐

caused damage: ☐ low ☐ medium ☐ high

Causes and successful countermeasures:

Comments on distributed process support and process documentation:

Page 14:

2. Requirements Engineering

Which challenges did emerge in this phase during distributed software development?

☐ There were **no specific challenges** in this area, which are due to distributed software development.

☐ There were challenges in this area, which are due to distributed software development, **but I do not know them**.

☐ There were the following challenges:

Page 15:

Copyright © Arbeitsgruppe Software Systems Engineering Heidelberg

1.)____

never	rarely	often	mostly	always
<5 %	5-25%	25-75	75-95	>95%
☐	☐	☐	☐	☐

caused damage: ☐ low ☐ medium ☐ high

Causes and successful countermeasures:

2.)____

never	rarely	often	mostly	always
<5 %	5-25%	25-75	75-95	>95%
☐	☐	☐	☐	☐

caused damage: ☐ low ☐ medium ☐ high

Causes and successful countermeasures:

3.)____

never	rarely	often	mostly	always
<5 %	5-25%	25-75	75-95	>95%
☐	☐	☐	☐	☐

caused damage: ☐ low ☐ medium ☐ high

Causes and successful countermeasures:

Copyright © Arbeitsgruppe Software Systems Engineering Heidelberg

Comments on distributed Requirements Engineering:

Page 16:
3. Architectural design and implementation

Which challenges did emerge in this phase during distributed software development?
☐ There were **no specific challenges** in this area, which are due to distributed software development.
☐ There were challenges in this area, which are due to distributed software development, **but I do not know them**.
☐ There were the following challenges:

Page 17:

1.)____

	never	rarely	often	mostly	always
	<5 %	5-25%	25-75	75-95	>95%
	☐	☐	☐	☐	☐

caused damage: ☐ low ☐ medium ☐ high

Causes and successful countermeasures:

2.)____

	never	rarely	often	mostly	always
	<5 %	5-25%	25-75	75-95	>95%

Copyright © Arbeitsgruppe Software Systems Engineering Heidelberg

☐ ☐ ☐ ☐
☐

caused damage: ☐ low ☐ medium ☐ high

Causes and successful countermeasures:

3.)______________________________________

never	rarely always	often	mostly
<5 % >95%	5-25%	25-75	75-95
☐	☐ ☐	☐	☐

caused damage: ☐ low ☐ medium ☐ high

Causes and successful countermeasures:

Comments on distributed architectural design and implementation:

Page 18:
4. Testing

Which challenges did emerge in this phase during distributed software development?
☐ There were **no specific challenges** in this area, which are due to distributed software development.
☐ There were challenges in this area, which are due to distributed software development, **but I do not know them**.
☐ There were the following challenges:

Copyright © Arbeitsgruppe Software Systems Engineering Heidelberg

Page 19:

1.)____

never	rarely	often	mostly	always
<5 %	5-25%	25-75	75-95	>95%
☐	☐	☐	☐	☐

caused damage: ☐ low ☐ medium ☐ high

Causes and successful countermeasures:

2.)____

never	rarely	often	mostly	always
<5 %	5-25%	25-75	75-95	>95%
☐	☐	☐	☐	☐

caused damage: ☐ low ☐ medium ☐ high

Causes and successful countermeasures:

3.)____

never	rarely	often	mostly	always
<5 %	5-25%	25-75	75-95	>95%
☐	☐	☐	☐	☐

caused damage: ☐ low ☐ medium ☐ high

Causes and successful countermeasures:

Copyright © Arbeitsgruppe Software Systems Engineering Heidelberg

Comments on distributed testing:

____
____
____
____

Page 20:

□ **Yes, I would like to receive the survey results by email.**
My email address: ____
(Your email address will be used for this purpose only and will be deleted after the mailing of the survey results.)

<u>**Thank you for completing the questionnaire!**</u>

Copyright © Arbeitsgruppe Software Systems Engineering Heidelberg

Dokument Information

Titel	The Challenges of Distributed Software Engineering and Requirements Engineering: Results of an Online Survey
Datum	28.09.2007
Version	1.0
Status	Draft
Verteilung	http://www-swe.informatik.uniheidelberg.de/research/publications/reports.htm

Copyright 2005, Arbeitsgruppe Software Systems Engineering, Heidelberg
Alle Rechte vorbehalten. Diese Veröffentlichung darf für kommerzielle Zwecke ohne vorherige schriftliche Erlaubnis des Herausgebers in keiner Weise, auch nicht auszugsweise, insbesondere elektronisch oder mechanisch, als Fotokopie oder als Aufnahme oder sonst wie vervielfältigt, gespeichert oder übertragen werden. Eine schriftliche Genehmigung ist nicht erforderlich für die Vervielfältigung oder Verteilung der Veröffentlichung von bzw. an Personen zu privaten Zwecken.

Document Information

Title	The Challenges of Distributed Software Engineering and Requirements Engineering: Results of an Online Survey 45
Date	28.09.2007
Version	1.0
Status	Draft
Distribution	http://www-swe.informatik.uniheidelberg.de/research/publications/reports.htm

Copyright 2005, Software Systems Engineering, Heidelberg
Copyright 2005, Software Systems Engineering, Heidelberg
All rights reserved.

This publication, whether the whole or part of the material is concerned, may not be commercially photocopied, reproduced, distributed in electronical, mechanical or any other form, stored in data bases or translated without previous written permission from the authors. For the reproduction or distribution of the publication for private purposes no written consent is required.

Copyright © Arbeitsgruppe Software Systems Engineering Heidelberg