Architectural Design and Documentation: Waste in Agile Development?

Christian R. Prause
Fraunhofer FIT
Schloss Birlinghoven
Sankt Augustin, Germany
prause@acm.org

Zoya Durdik
Research Center for IT (FZI)
Haid-und-Neu-Straße 10-14
Karlsruhe, Germany

***Abstract*—There is a problem with documentation and architectural design in agile projects. This is the result of interviews we have conducted with 37 software engineering experts from industry and academia. In our paper, we analyze the interview results and the origins of the revealed issues. We propose ways to integrate software design methodologies into agile development, and reputation mechanism to solve documentation problems. The contributions of this paper are (i) an investigation of expert opinions on design and documentation problems, (ii) an analysis of problem origins, and (iii) proposals for future improvements of agile processes.**

***Keywords*-agile development; software design; documentation; software quality; reputation**

I. Introduction

Agile software development is increasingly applied in industry and is also a popular research topic in academia [1], [2]. "Agile" means an incremental approach to development with a strong focus on project goals and intensive customer involvement. It has lower risks, few management overhead and forwards a good team atmosphere [2]–[4].

Agile development avoids activities (commonly called *waste*) that do not directly contribute to project goals [5]. Documentation and architectural design activities are sometimes regarded as *waste*, as they do not contribute a direct value to the end-product [6]. However, both are important for project success, dissemination, maintenance and evolution [7]. Architectural design contributes to the quality of the system, supports reuse of architectural knowledge and cross-project reuse of other software artifacts, and supports system maintenance and evolution [8]. Similarly, although disliked by developers [9], documentation contributes to quality and maintenance: "internal documentation is one of the most overlooked ways of improving software quality and speeding implementation" [10]. Thus, agile development might serve as an excuse for not doing important but unpopular tasks [11]. We treat source code as a special form of documentation: it is not only a medium of communication between man and machine but also between humans. And as the only precise description of a system, it has a particularly important role [12].

So, is there a problem with documentation and architectural design in agile software development? If yes, where would it come from? And how could it be solved?

We conducted structured interviews with 37 software engineering experts (see Section II), and found the following results (Section III): (1) The experts believe that there is indeed a problem with documentation and architectural design in agile projects. (2) The same holds true for source code. (3) These problems are also relevant for non-agile projects to a certain degree.

Section IV analyzes origins of the revealed issues with a focus on agile development. In Section V we then propose to improve documentation with a reputation system, which might be well in tact with agile philosophy. Further on, we propose to introduce explicit architectural design into agile development. In Section VI we discuss threats to validity and future work. Finally, Section VII concludes the paper.

Our contributions are: (i) a structured investigation of expert opinions on the problem of documentation and architectural design in agile development, (ii) an analysis of possible origins of the problem, and (iii) motivation of future research, and presentation of prospective solutions to the identified problems.

II. Study Methodology

This section explains the organization of the survey and provides details on the experts' background.

A. Organization of Interviews

The survey was organized as part of the PhD Working Groups (PWG[1]) during the 8^{th} the European Software Engineering Conference and the ACM SIGSOFT Symposium on Foundations of Software Engineering 2011. We investigated adoption of agile software development in practice, its perceived benefits and drawbacks, future research demands, and quality aspects, in particular, the state of architectural design and documentation.

The PWG organizers selected structured interviews with experts as research method. A structured interview is a quantitative research method, which ensures that each interview consists of exactly the same questions. The interviews were supported through questionnaires in printed and electronic versions. Our questionnaire included questions with free-text and multiple choice answers, yet most multiple choice questions also allowed a free-text answer. The interviews took about 20 minutes and were divided into three parts. The first part contained general

[1] http://pwg.sed.hu/ — ESEC/FSE PhD Working Groups web site

questions about participant's background (e.g. experience, occupation, areas of professional interests, etc.), and the last part questions to evaluate the survey itself. The main part contained the actual questions about agile development.

B. Interviewees' Background

About ten tentative participants did not consider themselves as experts, and opted out of the survey. Three (out of 40) interviews were not completed. Out of the remaining 37 interviews, 31 ones were with experts mainly occupied in academia and 6 with industrial experts. Note, several experts that we counted as "mainly academic" were working in industry in parallel, or possessed prior industrial experience.

The different levels of professional experience were approximately evenly represented in the survey. While the positions of academics varied from PhD students to full professors, the industry group was represented through group leaders and senior developers. Interviewees mentioned management, theory and early phases of the software-development life cycle (SDLC) as their areas of professional interests ($\approx 15\%$ each), yet the actual software development and late SDLC were particularly interesting ($\approx 30\%$ each).

Most experts (65%) had used agile processes in the past and planned to use them in the future. About 20% had not used agile processes, but were planning to do so. Finally, about 12% of our subjects were not going to use agile processes in the future, and only one of the participants had used agile development in the past and would not use it again. Thus, the majority of the participants had a positive attitude towards and experience with agile development.

Scrum (49% had used it), open source development (32%) and Extreme Programming (XP) (22%) were the best known agile development methods. While open source development is not an agile method per se, it shares a number of principles and values, like customer involvement, blending design and implementation, reliance on highly motivated individuals, and continuous design and adoption [13].

III. Problems in Agile Development

This section discusses problems of agile development based on the results of the interviews.

A. Problems and the Essence of Agile Development

First, we asked the experts what they personally regard as the main disadvantage of agile development. Most experts suggested that agile methods could only be applied to projects that have certain characteristics (32%), followed by a lack of explicit design support (24%) and neglecting of documentation (15%). Other potential disadvantages did not gather significant support. Regarding the first disadvantage, agile and non-agile methods are best applicable in different environments [14]. This situation cannot be improved through process amendments to agile development but would require radically new approaches. Therefore, we focus on the second and third ranked problems in the later Sections IV and V. Additionally, only 14% of the experts, all from academia with less than 20 years of experience, agreed that a lack of formalities, such as documentation and architectural design, is the essence of agile development. Hence, the lack of formalities is not inherent to agile development.

Table I
Experts structured by particular problems (in percent)

	Source code	**Doc.**	**Design**
All	47%	74%	93%
by work perspective			
Academia	48%	68%	92%
Industry	40%	100%	100%
by area of interest in software development life cycle			
Early SDLC	67%	70%	90%
Late SDLC	47%	76%	100%
Development	56%	74%	89%
Management	57%	78%	83%
Theory	17%	66%	88%
by work experience			
< 5 years	43%	88%	100%
$5 - 10$ years	60%	82%	90%
$10 - 20$ years	33%	71%	100%
> 20 years	43%	40%	80%
by agile process experience			
Scrum	50%	65%	88%
Open source dev.	56%	72%	89%
Extreme Programming	50%	57%	100%

B. A Detailed Investigation of the Problems

Table I summarizes which experts agree that generally problems are existent. About 10% of the questions were answered with "don't know", and thus were not included. Experts working in academia and industry have similar opinions about problems with design and documentation. While problems with the quality of source code are not seen as the most pressing concern, both, experts from industry and academia, recognized problems with design and documentation. Experts interested in theory saw less documentation problems, and even less ones with the quality of source code (17%). Problems with the source code were mostly noted by experts interested in the early SDLC. Yet they all were aware of design and documentation problems.

In open source software development, the quality of code, as a medium of communication among distributed developers, might be more important. Hence, experts were more sensitive to code problems. Similarly, XP developers might experience less documentation problems because knowledge is spread by different means, e.g. pair-programming. XP is famous for putting rapid implementation above architectural design, which is clearly reflected in the experts' opinions.

Breaking down the results based on experience shows that there is a tendency that the more work experience participants had, the less problems with documentation they saw. We found correlations $r = 0.07$ for source code, $r = 0.34$ (significant, $p < 0.05$) for documentation in general, and $r = 0.18$ for architectural design. An explanation could be that experts move farther away from

Table II
REASONS STRUCTURED ACCORDING TO THE EXPERTS' VOTES AND EXPERTS' BACKGROUND (IN PERCENT)

Potential reasons	All experts	Academia	Industry	< 5 yrs.	$5-10$ yrs.	$10-20$ yrs.	> 20 yrs.	Management	Early SDLC	Development	Late SDLC	Theory
Some developers do not know how to do it	46	45	50	78	55	25	22	56	64	55	50	50
Some developers do not care enough for quality	49	52	33	100	64	13	11	56	64	65	56	75
Some developers do not have the time to invest due to other tasks	46	45	50	56	82	38	0	56	82	65	50	50
Documentation and design are not explicitly considered	46	52	17	44	27	88	33	33	46	40	50	50
Quality goals are not set	35	39	17	78	27	25	11	33	46	50	44	50
Individual developer has limited benefit from it himself	41	45	17	67	55	25	11	33	55	55	56	75
Other project goals are more important	19	19	17	11	27	13	22	22	36	25	22	13

implementation activities when advancing in their jobs, and therefore perceive less problems.

Additionally, we investigated opinions on the influence of explicit architectural design on software quality and the development process in general. The majority of the experts was convinced that explicit architectural design improves software quality and processes *slightly* (35%) or *significantly* (32%). An additional 22% were generally positive yet concerned about the effort to be spent on architectural design and artifacts. Only one industrial expert was skeptical, while three other ones did not answer this question.

IV. THE ORIGINS OF THE PROBLEMS

Section III-B reveals that the majority of experts perceives a lack of documentation and architectural design in agile projects, and attributes this problem neither to the intention, nor the essence of agile development. Therefore, we asked the experts about possible reasons for poor design, documentation and code. The answers are summarized in the Table II, structured according to the expert's background. The numbers are provided in percent.

According to our experts, the reasons for possible problems with design, documentation and code are that developers might not care about it (49% agree), do not know how to do it properly, lack time, do not explicitly consider documentation and design (46% each), have limited personal benefit (41%), and miss defined quality goals (35%). Minor reasons for poor documentation and neglected architectural design are that other goals conflict with the goal of writing good documentation (19%), projects follow a fast prototyping approach, linguistic and cognitive dissonance (inability to write well, express oneself), lacks of professional management and of the awareness of the importance of documentation, and frequent changes in requirements.

According to Table II, experts from industry see a lack of time and a lack of knowledge as important reasons, while other reasons are not deemed very important. Experts occupied in academia, however, consider most of the named reasons to be important. Two less important reasons, from the academic point of view, are that quality goals are undefined and that other project goals are more important.

Younger experts seem to be convinced that developers do not care enough for design and documentation, and mentioned undefined quality goals and lack of knowledge as possible reasons. Time concerns are recognized as the major reason by experts within 5 to 10 years experience. For senior experts, the main reason is that documentation and design are not explicitly considered. In general, more experienced interviewees accept less explanations for the existence of problems. This is in congruence with our finding that senior experts see fewer problems in general. And none of the most experienced experts mentioned lack of time as a reason.

Table III
MOST EFFECTIVE COUNTER-MEASURES ACCORDING TO THE EXPERTS

Solution methods evaluated by experts	%
Reviewing	60%
Separate quality control roles or department	22%
Pair-programming	6%
Agreement on common rules	6%
Other	6%

Experts interested in early SDLC and development consider insufficient time, lack of knowledge, not caring about and limited personal benefits as the main reasons for poor documentation and design. For experts interested in late SDLC and theory, carelessness and limited personal benefits are the main reasons. Experts interested in management believe that lack of knowledge, not caring about and insufficient time are more important reasons, however, they also mentioned other reasons.

We asked our experts if the reasons for design and documentation problems are specific to agile projects. About 88% of our experts rejected this statement. Thus, the reasons might be true for other development methods. Nevertheless, they are highly relevant in agile software development.

V. DEALING WITH DESIGN & DOCUMENTATION ISSUES

This section explores possibilities for improving design and documentation problems in agile development. We investigate the effectiveness of classical approaches and propose two solutions that might be worth further research.

A. Classical Approaches to the Problems

We asked our experts to chose from a list what countermeasures they see as most effective (see Table III). Methods based on enforcement (reviewing and dedicated roles)

scored high, and together collected the vast majority of votes (82%). Pair-programming and agreement on rules each collected a few votes, while explaining of benefits, rewards, and showing good examples were rarely deemed effective.

B. Can Reputation Systems Solve Documentation Problems?

Our experts consider reviewing as highly effective, and 47% of them think that new and agile ways of improving documentation should be researched more thoroughly. We have therefore inquired into reputation systems to improve source code quality and documentation in agile projects. According to the Oxford Dictionary, reputation is what is generally said about the abilities or qualities of somebody. It is sometimes considered as what makes human societies work, as it enables a society to deal with selfish individuals. A *reputation system* is a software that determines a means of a user's reputation from his actions. The computed reputation scores can then be published to cause a social effect. Reputation systems have become an important factor in online communities to alter user behavior and foster well-behaving. Instead of enforcing desirable behavior through tight management control, reputation systems motivate behavior by displaying good (or bad) reputation. The online market place eBay, for example, does not control if both parties involved in a transaction behave well. Instead, ratings are gathered to derive reputation scores, motivating users to behave well in order to achieve good reputations [12], [15]. Likewise, a reputation system could foster well-behaving in software projects. It would continuously analyze the contributions of developers — i.e. writing documentation or well-readable code — and derive and publish reputation scores (c.f. [16]). For instance, Dencheva et al. [17] showed that a reputation system can improve contributions to a corporate Wiki: Users are encouraged to rate articles. A reputation system then collects this feedback, combines it with authorship information, derives personal reputation scores, and publishes them.

About 15% of our experts doubted that reputation could work. Some experts were plainly skeptical because of a gut feeling. Others mentioned that there might be a reception and acceptance problem in company environments, making such solution more suitable for open source development. The number of skeptics was particularly high among the experts from industry (50%). They thought that reputation would not fit into an industry environment, and stated that it might lack enough force to cause a positive effect.

Yet the majority of experts (85%) was convinced that a reputation system could indeed be beneficial for documentation. However, 30% argued it could be complicated to realize in a good way. They reasoned that reputation would only motivate people based on peer pressure, and that its effectiveness was therefore hampered by the diversity of the developers' group. As a side effect, a reputation system might make people feel guilty. One expert mentioned that a reputation system would probably motivate others but not himself. Nevertheless, 85% of the experts saw future research in this direction as promising. Among them, 70% (or 60% of all experts) supported that a reputation system will have a positive effect on documentation in agile projects by motivating "good" (pro-social) behavior.

Table IV
ARCHITECTURAL DESIGN PRACTICES APPLIED BY OUR EXPERTS

Used architectural practice	**%**	**Experience**
A continuous iterative design process embedded into agile development, artifacts are updated regularly	30	10 to 20 years
Agile development is not used	24	About 10 years
Some initial design, however architectural artifacts are not updated later	16	Mixed
No explicit design iteration	11	About 20 years
Type of the projects does not require architectural design (repeating simple projects)	5	10 to 20 years
Other: Design meeting followed by individual implementation	3	$<$ 10 years
No answer provided	11	About 10 years

C. Towards Explicit Architectural Design

There are two ways to improve architectural design in agile methods [18]: *Agile architectural modeling* makes architectural modeling more agile by lowering its overhead. This could be achieved by using an incremental customer-involved process. Challenges here lie in the definition of architectural refactoring and a suitable redefinition of the term overhead. *Agile methods with architectural modeling* introduce architectural modeling into a particular agile method, such as Scrum. In this scenario, an agile process works with additional artifacts (i.e. architectural models and documentation). The benefits of architectures would need to be exploited in order to support an overall agility, e.g. use of architecture for cost estimation, early evaluations of architectural design decisions and the use of architecture to elicit the right requirements.

The second scenario can be further refined into several sub-scenarios. *"Iteration Zero" approach*: An initial vision of the system including initial design is created during the first iteration of the development. Architectural design is more a draft that is changed during later development. *Extended "Iteration Zero" approach*: There are several iterations for designing the system, thus a more detailed design followed further on is created. And *continuous iterative design*: Design is embedded into agile development, and architectural artifacts are updated regularly.

To verify which approaches might be feasible in praxis, we have asked the experts if and how they integrated architectural design in their agile projects. The results are summarized in the Table IV. Almost half of the experts practice one of the architectural design approaches described above, and the majority (89%) believed explicit architectural design is beneficial. Still, 11% of the experts did not use any of the proposed practices, and about 5% of the participants were not involved in projects that would require architectural design. The participants not using

design approaches also stated that architectural design might only lead to some quality and process improvement, and in particular that it is unclear if it can be compared to the effort spent on architectural design and artifacts. We would like to conduct a more detailed investigation of quality problems compared to the way of architectural design integration into the development process.

VI. Threats to Validity and Future Work

The data presented in this paper reports on the results of structured interviews with 37 experts (15% of the participants of ESEC/FSE 2011), and therefore reflects their opinions, i.e. how they perceive agile development. This perception does not necessarily reflect the actual situation in all agile projects, and our data should be treated with care. Although industrial experts are underrepresented, our academic experts often worked closely with industry, had an industrial past, or had practical experience in agile projects.

In the future, we would like to improve the questionnaire based on our experiences with the first version. We are planning to repeat the survey with a larger group of participants, and increase the participation of industrial experts.

VII. Conclusion

In this paper we presented the results of structured interviews we conducted on the topic of architectural design and documentation in agile development. The interviewed experts had the opinion that there is indeed a problem with both. We provided a detailed analysis of our results and investigated origins of the revealed issues. Further on, we proposed reputation as a way to improve documentation in agile development, and to introduce an explicit architectural design. We also provided an evaluation of these solutions based on the experts' opinions. While the data presented in this paper reflects personal opinions, we interviewed about 15% of the participants of a solid software engineering conference. Thus, we believe that our study represents founded opinions and provides valuable insights into the topic. However, most experts also believed that the problems are not specific to agile development. They stated that the topic, and proposed solutions, are worth future research.

We are planning to repeat the survey with a larger group of participants, involving more experts from industry.

Acknowledgments

We are grateful to ESEC/FSE PWG organizers (Arpad Beszedes, István Siket, Lajos Schrettner and Laurie Williams) and our experts for sharing their time and expertise with us. This work was supported by German BMBF, grant No. 01BS0822, and ebbits EU project No. 257852.

References

[1] M. A. Babar, "An exploratory study of architectural practices and challenges in using agile software development approaches," in *WICSA/ECSA 2009.*, 2009, pp. 81 – 90.

[2] O. Salo and P. Abrahamsson, "Agile methods in european embedded software development organisations: a survey on the actual use and usefulness of xp and scrum," *Software, IET*, vol. 2, pp. 58 – 64, 2008.

[3] B. Sheth, "Scrum 911! using scrum to overhaul a support organization," in *Agile Conference*, 2009, pp. 74 – 78.

[4] T. Dingsøyr, G. K. Hanssen, T. Dybå, G. Anker, and J. O. Nygaard, "Developing software with scrum in a small cross-organizational project," in *LNCS*, vol. 4257, 2006, pp. 5–15.

[5] P. Abrahamsson, M. Babar, and P. Kruchten, "Agility and architecture: Can they coexist?" *IEEE Software*, vol. 27, 2010.

[6] R. Reussner and M. Grund, "Industrialisierung des software-engineering – architektur und agilität," *VKSI*, vol. 2, 2010.

[7] E. Tryggeseth, "Report from an experiment: Impact of documentation on maintenance," *Emp. Softw. Eng.*, vol. 2, 1997.

[8] P. Clements, F. Bachmann, L. Bass, and D. Garlan, *Documenting Software Architectures: Views and Beyond*. Addison Wesley Longman, 2002.

[9] B. Selic, "Agile documentation, anyone?" *IEEE Software*, vol. 26, no. 6, pp. 11–12, Nov/Dec 2009.

[10] J. Raskin, "Comments are more important than code," *ACM Queue*, vol. 3, no. 2, pp. 64–62, 2005.

[11] S. Rakitin, "Manifesto elicits cynicism," *IEEE Computer*, vol. 34, no. 12, p. 4, December 2001.

[12] C. R. Prause, "Reputation-based self-management of software process artifact quality in consortium research projects," in *ESEC/FSE 2011*, 2011, pp. 380–384.

[13] R. Goldman and R. P. Gabriel, *Innovation Happens Elsewhere: Open Source as Business Strategy*. Morgan K., 2005.

[14] B. Boehm, "Get ready for agile methods, with care," *IEEE Computer*, vol. 35, pp. 64–69, 2002.

[15] A. Jøsang, R. Ismail, and C. Boyd, "A survey of trust and reputation systems for online service provision," *Decision Support Systems*, vol. 43, no. 2, pp. 618–644, 2007.

[16] C. R. Prause and S. Apelt, "An approach for continuous inspection of source code," in 6^{th} *WoSQ*, 2008.

[17] S. Dencheva, C. R. Prause, and W. Prinz, "Dynamic self-moderation in a corporate wiki to improve participation and contribution quality," in 12^{th} *ECSCW*, 2011, pp. 1–20.

[18] Z. Durdik, "Towards a process for architectural modelling in agile software development," in *7th Int. Conf. on the Quality of Software Architectures. QoSA.*, 2011.