Proceedings of the Federated Conference on Computer Science and Information Systems pp. 975–978
DOI: 10.15439/2018F151
ISSN 2300-5963 ACSIS, Vol. 15

Problems and Solutions of Software Design in Scrum Projects

Jakub Miler
Gdansk University of Technology
Faculty of Electronics, Telecommunications and Informatics
11/12 Narutowicza St., 80-233, Gdansk, Poland
Email: jakub.miler@eti.pg.edu.pl

Kamil Kajdy
IHS Global sp. z o.o.
163 Marynarki Polskiej St.,
80-868, Gdansk, Poland
Email: kamil.kajdy@gmail.com

***Abstract*— The aim of the paper is to identify the problems and solutions of the software design in Scrum project as well as to analyze the effectiveness of the solutions. Through a series of workshops with 4 experts from IT industry and academia we have identified 52 problems and 99 unique solutions. In this paper we present a list of 10 common problems and 5 solutions for each problem selected by the number of sources. The effectiveness of the solutions to the given problems was evaluated in an opinion survey by 39 respondents with experience both in software design and in the Scrum framework. This evaluation provided for our initial recommendations on the choice of solutions to particular problems.**

I. Introduction

Software design is one of the key elements of software engineering [1], [2]. Systematic approach to architecture, code structure, data processing, and other aspects is required for many types of systems based on their size, complexity, distribution, and quality factors e.g. safety, security [3], [4]. Development practices such as pair programming, continuous integration, test driven development [5], [6], design patterns, refactoring [7] or clean code principles [6], [8] provide solutions to many problems, but their application in practice is challenged by the development methodology, technology, team, customer and many more.

Scrum defines only the roles, artifacts and events of the development process on a general level and leaves the room for specific decisions and actions to the Scrum Team [9]. This includes the design, programming and testing of software, where the Scrum Team should be multifunctional to cover all the competencies necessary to deliver the product [10] and include the role of a software architect if necessary [11]. Additionally, Scrum promotes working product increment after each sprint leaving little time for detailed approach to architecture and design [6], [10]. It is recommended to design as little and as late as possible to avoid negating the design by changing requirements [12], [13], [14]. This approach results in increasing technical debt which is related to the low quality of design and code [12]. It is also not possible to apply in case of complex systems and scaled Scrum [15].

Some of the recent research studied the relationship between the architecture-centric design and the agile development, but the authors focused either on the eXtreme Programming framework [16] or the agile projects in general [17]. This shows that the integration of the software design principles and the Scrum framework is not straightforward and calls for a detailed inquiry.

Our research goal was to analyze the problems and solutions of software design in Scrum projects. To achieve this goal, 3 research questions were formulated: (RQ1) What are the problems with software design in the Scrum projects? (RQ2) What are the solutions to the software design problems in the Scrum projects? (RQ3) Which solutions to the problems with software design can be recommended to the Scrum projects?

The contribution of this paper is the identification of the problems and solutions of software design in Scrum projects as well as some initial recommendations of the effective solutions to the most common problems.

The paper is organized as follows. Section II describes the research method, the workshops with experts and the online survey. Section III presents the list of the top problems and their solutions as well as the evaluation of these solutions together with some recommendations. Section IV discusses threats to the validity of this research followed by the conclusions in Section V.

II. Research Method

Our research method comprised two techniques: the workshops with experts to identify the problems and their solutions, and the online survey to evaluate the perceived effectiveness of the solutions to particular problems.

The workshop was designed as a structured multi-phase brainstorming session with the following steps:

1. introduction to the workshop, explanation of the goals and the scope,
2. individual identification of problems,

This work was supported by DS Funds of ETI Faculty, Gdansk University of Technology.

3. discussion of the problems identified in step 2, aggregation of duplicate problems,
4. individual identification of the solutions to the problems resulting from step 3 (a solution may solve more than one problem),
5. discussion of the solutions identified in step 4, aggregation of duplicate solutions.

The workshop involved a domain expert and one of the researchers (K. Kajdy) as a moderator. The scope focused on the specific aspects of the Scrum agile framework: development in short iterations, changing requirements, self-organized teams, and little documentation. Additionally, the aspects of software design were restricted to the following: component integration, architecture, design patterns, NoSQL or relational databases, user interfaces, modularization, and refactoring.

We have carried out workshops with 4 experts with at least 2 years of experience in both software design and the Scrum framework. The experts played the roles of developers and/or Scrum Masters. Each workshop resulted in a distinct list of problems and solutions to these problems. Finally, a compiled list of problems and their solutions was built from the results of all 4 workshops. The merging was based on keyword analysis.

We have selected 10 problems and 5 solutions to each of these problems for the evaluation survey (50 solutions in total) to limit the size of the survey and increase the rate of feedback. The problems and solutions were selected primarily based on the total number of indications in the source workshops.

The effectiveness of each solution in relation to a given problem was assessed in a Likert-type 5 level scale of 1 to 5, where 1 meant "a solution is totally ineffective to the problem" and 5 meant "a solution is very effective to the problem" with an escape answer "I don't know". We have also asked about the respondents' experience in software design and in the Scrum framework. Although Likert-type scale is ordinal, in the data analysis we have treated it as numerical with assigned values of 1 to 5. The evaluation of each solution's effectiveness was calculated as a weighted average, where weights represented the respondents' experience: 0.1 – under 1 year; 0.3 – 1-2 years; 0.6 – 2-3 years; 0.85 – 3-5 years; 1 – above 5 years.

III. Results

The identification workshops were carried out in May and June 2017. On average, the experts identified 25.75 problems, 31 unique solutions and 75.75 total solutions per workshop. In total, they have identified 52 problems, 99 unique solutions and 231 total solutions to all problems. The detailed results of the workshops as well as the full list of merged problems and solutions are available in [18].

The evaluation survey was carried out in August and September 2017 with Google Forms. It was promoted among the IT practitioners via e-mail and social media. In total, 39 respondents took part in the survey. 22 respondents (56%) had at least 2 years of experience with software design and 20 respondents (51%) had at least 2 years of experience with Scrum.

Table I and Table II present the identified problems and their evaluated solutions. The columns are as follows: identifier, problem/solution name, evaluation with a weighted average and weighted standard deviation in parentheses, survey sample size (N), and number of indications in the workshops (n). The problems are ordered by the number of indications (n) and the solutions for each problem are ordered by their evaluation (avg.).

Most of the top evaluated solutions reached a score close to 4 or more than 4. Problem P2 is the exception with a top evaluated solution of 3.49. This indicates the need for further research on its better solutions. Problems P3, P4, P5, P7, P8, and P10 can be assigned a clear leading solution with top score of more than 4. Additionally, more than one solution for problems P7 and P8 have reached the score of 4. The top scoring solutions for problems P1, P6, and P9 have an evaluation of slightly below 4, but the top solutions are still significantly ahead of the rest except for the problem P6, where 4 top solutions are enclosed within the range of 0.1.

As for the lowest scoring solutions, it can be observed that the solutions S1 and S36 are evaluated as least effective for all problems they were assigned to (S1 to problems P1, P7, P9, and P10; S36 to problems P5 and P6) with sample size of more than 30. They were, however the top solutions in the identification phase resulting from 4 and 3 sources respectively. The experts' belief in their effectiveness has been significantly challenged by the survey. It may indicate that these solutions strongly depend on factors specific to business environments (e.g. personnel, culture, type of products), which can be further studied in future research.

It should be noted that some of the solutions can be hard to apply in a strictly agile environment. Formal review of projects (S4) can go beyond the visibility and transparency principles of Scrum and Agile Manifesto leading to an overly monitored and manually managed team. Task estimation and accounting recommendations such as S25 or S31 can also hamper the customer-developer trust Agile is based on. S34 refers to the role of a project manager, which is outside of the Scrum framework and calls for a project management methodology on top of Scrum. This can be considered a non-agile practice.

Some solutions are also technology or architecture dependent e.g. NoSQL databases (S24), API versioning (S52) or microservices (S61), which also limits their application. It may not be beneficial or possible at all to implement such solutions in particular systems.

The proposed list of problems and their evaluated solutions has mostly educational use by Scrum developers, Scrum Masters and coaches. The application of a solution in a particular project shall always be discussed and accepted within the Scrum Team.

TABLE I.
PROBLEMS P1-P5 AND THEIR EVALUATED SOLUTIONS

Id	Name	Avg.	N	n
P1	**Team work assessed mainly with of code increments and new functionalities**			**8**
S3	Promoting quality and designing in the organization and to the client	3.88 (1.35)	34	2
S2	Avoiding creating fast and large increments at the expense of design and quality of code	3.53 (1.34)	36	2
S4	Formal review of projects	3.20 (1.39)	32	2
S5	An organization's policy that only part of the time is devoted to working with the code	3.18 (1.10)	34	2
S1	Professional Scrum Master teaching team communication and promoting issues of architecture and design at the meetings	3.12 (1.38)	33	4
P2	**Problems with expanding and modifying the production database in the client's environment**			**6**
S17	Automation of creating data models from code	3.49 (1.45)	32	1
S24	NoSQL databases	3.05 (1.49)	18	1
S21	Designing the database changes one sprint earlier or at the very beginning of the sprint	2.96 (1.19)	31	1
S20	Small database design at the beginning (the less data collected, the less data to update)	2.78 (1.48)	30	1
S16	Making modifications to the database once every few sprints	2.61 (1.22)	30	3
P3	**Recognizing refactoring as an increment by the client, despite the client's resistance**			**5**
S28	Doing refactoring partially in each sprint, not all in one sprint	4.04 (1.21)	38	1
S29	Using the refactoring automation tools	3.87 (1.02)	29	1
S25	Including the refactoring costs in the price of an expensive task	3.77 (1.08)	33	5
S27	Educating the client and obtaining approval for corrective and maintenance actions	3.50 (1.12)	36	2
S26	Introduction of stabilization sprints for code maintenance	3.36 (1.30)	33	2
P4	**Improperly defined tasks that hamper planning and design**			**5**
S30	Grooming before planning - examining and presenting details of a given User Story	4.09 (0.77)	33	3
S31	Overestimating tasks to leave time for "unpredictable"	3.89 (1.06)	36	1
S32	A business analyst present on the planning and available to the team	3.73 (1.09)	33	1
S34	Project Manager that accurately defines the tasks	3.71 (1.01)	36	1
S35	Behavior Driven Development - Gherkin language	3.07 (1.10)	17	1
P5	**Difficulties with introducing new functionalities due to architectural errors**			**5**
S42	Separation of views, data and business logic	4.50 (0.64)	35	1
S38	Applying the initial conceptual and design phase before the actual implementation	3.97 (0.88)	36	2
S39	A team using design patterns, standards, diagrams	3.90 (1.01)	33	2
S37	Making the client aware of the time needed for the design and that it will pay back	3.82 (1.18)	35	2
S36	Preparation of prototypes and preliminary design in "sprint 0"	3.33 (1.14)	35	3

TABLE II.
PROBLEMS P6-P10 AND THEIR EVALUATED SOLUTIONS

Id	Name	Avg.	N	n
P6	**Problems resulting from the selection of project technology in advance, before the implementation**			**3**
S38	Applying the initial conceptual and design phase before the actual implementation	3.98 (0.95)	35	2
S48	Careful selection of technologies - proven technologies for large projects, experiments with fast Proof of Concepts	3.94 (1.08)	36	1
S44	Checking the technologies available on the market as part of the task of the increment	3.91 (1.02)	34	2
S37	Making the client aware of the time needed for the design and that it will be pay back	3.88 (0.98)	35	2
S36	Preparation of prototypes and preliminary design in "sprint 0"	3.30 (1.07)	36	3
P7	**Problems with developing a uniform communication interfaces between modules**			**3**
S52	API versioning	4.37 (0.76)	32	1
S39	A team using design patterns, standards, diagrams	4.10 (0.95)	31	2
S3	Promoting quality and designing in the organization and to the client	3.90 (0.90)	32	2
S51	The design created 1 sprint earlier or at the very beginning of the sprint	3.37 (1.15)	33	1
S1	Professional Scrum Master teaching team communication and promoting issues of architecture and design at the meetings	2.90 (1.18)	31	4
P8	**Problems with technological debt and poor quality due to rush in implementation**			**3**
S54	Applying SOLID practices and adhering to the rules of clean code	4.37 (0.77)	33	1
S56	Multiphase code reviews	4.15 (0.89)	35	1
S28	Doing refactoring partially in each sprint, not all in one sprint	4.04 (1.18)	37	1
S27	Educating the client and obtaining approval for corrective and maintenance actions	3.85 (1.01)	36	2
S26	Introduction of stabilization sprints for code maintenance	3.75 (1.29)	35	2
P9	**Difficulties with breaking down tasks into smaller tasks**			**3**
S60	Transferring detailed design problems to separate meetings of selected people	3.94 (0.86)	36	1
S25	Including the refactoring costs in the price of an expensive task	3.58 (1.19)	30	5
S61	Application architecture based on microservices	3.38 (1.06)	29	1
S39	A team using design patterns, standards, diagrams	3.34 (0.97)	34	2
S1	Professional Scrum Master teaching team communication and promoting issues of architecture and design at the meetings	2.94 (1.20)	32	4
P10	**Mutual blocking of implementation tasks**			**3**
S30	Grooming before planning - examining and presenting details of a given User Story	4.16 (0.85)	35	3
S50	Informal conversations and arrangements (helping to avoid blocking tasks)	3.83 (1.05)	37	1
S39	A team using design patterns, standards, diagrams	3.71 (0.84)	32	2
S62	Assigning tightly related tasks to one developer	3.61 (0.95)	37	1
S1	Professional Scrum Master teaching team communication and promoting issues of architecture and design at the meetings	2.93 (1.24)	34	4

IV. Validity threats

A. Threats to construct and internal validity

We have controlled the workshop moderator's bias and his impact on experts with the structure of the workshop, which included the steps of individual identification (steps 2 and 4). Only then were the identified problems and solutions discussed and merged. The moderator was open to further expert's explanations.

The incorrect interpretation of the output from experts was controlled by writing down the output on the post-it notes and then discussing it to clearly understand the experts intentions. The moderator preserved all post-it notes after the workshop and built the resulting list of problems and solutions directly after the workshop referring to the notes and his fresh memory. For details on the workshop design see section II of the paper.

The interview experts represent the above average experience of our sample. Only 7 of 39 survey respondents had more experience, which puts the interview experts in the top quartile of the survey sample. What is the most important, the interview experts were able to identify large number of problems and solutions from their experience.

Our weight system is arbitrary at the moment, but it was designed to represent the assumed learning curve of software design and the Scrum framework based on the university syllabus and the authors' work experiences. We plan to study the learning curve of the Scrum framework in the future.

B. Threats to external validity

The number of experts was limited to 4 due to several factors. First, the set of data collected after 4 workshops was very satisfactory and we agreed on finishing this phase of research at this stage. Second, we required our experts to have experience both in software design and in the Scrum framework, which significantly limited the available sample. We have involved experts from various business environments: academia, technological start-up, small company, and large multinational corporation. The number of respondents was also limited due to the specific set of competencies required for the survey.

Our sample is not statistically random, but the experts and respondents were identified and contacted with various channels such as personal contacts, business contacts, social media, and recommendations from identified experts. This provided for a reasonably diverse group of practitioners.

We have based our research on data from experts and respondents working in the Polish market. This forms the natural limitation to our current results.

V. Conclusion

We have carried out 4 workshops with IT practitioners and identified 52 unique problems and 99 unique solutions. We believe that these results form a valuable answer to the research questions RQ1 and RQ2. Due to the limitations of this paper, we have presented only the 10 most commonly indicated problems as well as 5 solutions per problem selected for the survey.

We have acquired some evaluation of the effectiveness of the solutions to the 10 selected problems. We could point out some of the highly evaluated solutions as our initial recommendations as well as indicate the lowest evaluated solutions as risky. It should be considered, however, that the evaluations are based on the opinion poll only. This provides only a preliminary answer to the research question RQ3. Further and more detailed verification of the solutions' effectiveness in practice requires careful observation of a number of projects and can be done in future research.

Acknowledgment

The authors thank all the experts and respondents who took part in the identification workshops and the survey.

References

[1] I. Somerville, *Software Engineering*, 10th edition, Pearson, 2015
[2] R. S. Pressman, *Software Engineering: A Practitioner's Approach*, 8th Edition, McGraw-Hill Education, 2014
[3] J. Valacich, J. George, *Modern Systems Analysis and Design*, 8th edition, Pearson, 2016
[4] L. Maciaszek, *Requirements Analysis and Systems Design*, 3rd edition, Pearson Education Canada, 2007
[5] K. Beck, C. Andres, *Extreme Programming Explained: Embrace Change*, 2nd edition, Addison-Wesley, 2004
[6] M. Lacey, *The Scrum Field Guide: Practical Advice for Your First Year*, Addison-Wesley Professional, 2012
[7] R. C. Martin, *Agile Software Development, Principles, Patterns, and Practices*, Pearson, 2002
[8] R. C. Martin, *Clean Code: A Handbook of Agile Software Craftsmanship*, Prentice Hall, 2008
[9] K. Schwaber, *Agile Project Management with Scrum*, Microsoft Press, 2004
[10] K. Schwaber, J. Sutherland, *The Scrum Guide. Rules of the Game*, Scrum.org, 2017
[11] M. Cohn, *Succeeding with Agile: Software Development Using Scrum*, Addison-Wesley, 2010
[12] K. S. Rubin, *Essential Scrum: A Practical Guide to the Most Popular Agile Process*, Addison-Wesley Professional, 2012
[13] J. Rasmusson, *The Agile Samurai: How Agile Masters Deliver Great Software*, Pragmatic Bookshelf, 2010
[14] J. Sutherland, J. J. Sutherland, *Scrum: The Art of Doing Twice the Work in Half the Time*, Currency, 2014
[15] J. Diaz, J. Garbajosa, J. Perez, A. Yague, *Bridging User Stories and Software Architecture: A Tailored Scrum for Agile Architecting*, Agile Software Architecture: Aligning Agile Processes and Software Architectures, M. Ali Babar, A. W. Brown, I. Mistrik (eds.), Morgan Kaufmann, 2013
[16] R. L. Nord and J. E. Tomayko, "Software architecture-centric methods and agile development", *IEEE Software*, vol. 23, no. 2, pp. 47-53, 2006, DOI: 10.1109/MS.2006.54
[17] C. R. Prause and Z. Durdik, "Architectural design and documentation: Waste in agile development?", *2012 International Conference on Software and System Process (ICSSP)*, Zurich, 2012, pp. 130-134. DOI: 10.1109/ICSSP.2012.6225956
[18] K. Kajdy, *Analysis of software design in Scrum projects*, MSc Thesis, supervisor J. Miler, Gdansk University of Technology, Poland, 2017 (in Polish)