The Disappearance of Technical Specifications in Web and Mobile Applications

A Survey Among Professionals

Theo Theunissen(✉) and Uwe van Heesch

HAN University of Applied Sciences, Arnhem, The Netherlands
theo.theunissen@gmail.com, uwe@vanheesch.net

Abstract. In recent years, we have been observing a paradigm shift in design and documentation practices for web and mobile applications. There is a trend towards fewer up-front design specification and more code and configuration-centric documentation. In this paper we present the results of a survey, conducted with professional software engineers who build web and mobile applications. Our focus was on understanding the role of software architecture in these applications, i.e. what is designed up-front and how; which parts of the architecture are reused from previous projects and what is the average lifetime of such applications. Among other things, the results indicate that free-text design specification is favored over the use of modeling languages like UML; architectural knowledge is primarily preserved through verbal communication between team members, and the average lifetime of web and mobile applications is between one and five years.

1 Introduction

In recent years, we have been observing a paradigm shift in the software engineering community. Professional software development projects traditionally relied on upfront planning and design, distinct software phases and often a clear separation of roles and responsibilities within project teams. Ever growing time-to-market constraints, however, leads to high innovation pressure, which brought forth methods and techniques like agile project management, continuous delivery, and DevOps, which break with the traditional way of approaching software projects. Apart from this, primarily for web and mobile application development, developers now need to deal with an increasingly heterogeneous tool and language stack. In that domain in particular, software is often designed ad-hoc, or at best by drawing informal sketches on a white board while discussing with peers. The reasons for this are multifold: where applications must be developed and rolled out quickly, designers do not seem to see the value of spending much time on modeling and documenting solutions. A second reason is that software engineering has no guidelines for efficient modeling of heterogeneous multi-paradigm applications. This is partly due to the fact that software engineering curricula at universities are still heavily focused on traditional object-oriented analysis and design using UML, which is not a good fit for applications that are not

B. Tekinerdogan et al. (Eds.): ECSA 2016, LNCS 9839, pp. 265–273, 2016.
DOI: 10.1007/978-3-319-48992-6_20

purely object-oriented. In this paper, we describe a questionnaire-based survey, conducted to understand current design and documentation practices that companies use for developing web and mobile applications. Our investigation leads to the conclusion that in the design phase, "experimentation" with *proof of concepts* has the highest score, and "free-text documentation" is more used than technical documentation (UML, ERD). The continuity of knowledge is primarily achieved using free-text documentation and verbal communication.

The study is part of a larger research project, in which we develop an architecture framework that is streamlined for modern web and mobile applications. In the framework, we plan to provide just-enough architecture and design specification for supporting agile web and mobile application development, while preserving core decisions and design for re-use in subsequent projects.

The rest of this paper is organized as follows. Section 2 describes the research questions and presents and motivates the study design. Section 3 presents the results of the questionnaire and our interpretations with respect to the research questions. The next section presents potential threats to validity. Finally, we conclude and present directions for future work.

2 Study Design

In this section, we present our research questions and the study design.

2.1 Research Questions

As described in the introduction, the goal of our research project is to develop an architecture framework that optimally supports software engineers in building and maintaining web and mobile applications. The study presented was conducted to settle the baseline process and to get a better understanding of current design and documentation processes, as well as developers' concerns in the industry. In particular, we address the following research questions:

RQ1. How are web and mobile applications designed and how is design knowledge preserved in the industry?
RQ2. Which parts of the software architecture are re-used across web and mobile applications within a development team?
RQ3. What is the average life expectancy of web and mobile applications?

The first question aims at finding out how web and mobile applications are designed, i.e. which modeling languages, or more generically: which approaches, are used to support the design process. By design, we refer to activities conducted prior to technical implementation. The second objective of RQ1[1] is to find out how architectural knowledge about these applications is maintained. This includes decisions made during the architecting process, as well as information about the problem and solution space.

[1] Because of space limitations, in this paper, we bundled two of our original research questions into one.

RQ2 originates from the conjecture that development teams partially reuse architectural design from previous projects. Architectural reuse improves development efficiency, which contributes to fast time-to-market of features. Furthermore, reuse is a means for risk mitigation [1]. Here, we want to find out which parts of an architecture are typically re-used.

The third question concerns the timespan, developers expect their applications to reside on the market before they are discarded or subject to a major re-engineering. This is relevant because the cost-effectiveness of documentation effort is proportional to the expected lifetime of an application.

2.2 Methodology

A survey was conducted to collect data for our research questions. We chose to use a web-based questionnaire over individual interviews, because we wanted to reach a sufficiently large subject population. Questionnaire-based surveys additionally exhibit a higher degree of external validity than interviews [2]. When used with closed-ended questions and fixed response options, data gathered with questionnaires can easily be processed automatically. This is in contrast to interviews, which have high costs in terms of time per interview, traveling and processing the results. On the other hand, interviews provide greater flexibility and allow for more in-depth exploration of the respondents' answers. As described in Sect. 5, we plan to conduct interviews with a small number of the subjects at a later stage to get more in-depth insight in the phenomena we observe through the questionnaire.

We conducted a pilot study with three members from the target population to improve the wording, order and answering options of the questions.

2.3 Participants and Sampling

The target population of this study is professional software engineers who develop web and/or mobile applications. We used snowball sampling, i.e. the questionnaire was sent out to members from our professional network using e-mail. We asked the receivers to forward the questionnaire to colleagues and peers from their own network, which is a means to achieve a more randomized sample [3]. The questionnaire was spread in form of an online form.

3 Data Analysis and Interpretation

This section presents the data analysis and interpretation. We primarily use descriptive statistics to analyze the collected data. The section is divided according to the three research questions. Table 1 maps the research questions to the questions from the questionnaire.

A total of 73 subjects responded to the questionnaire. From these respondents, 39.7 % completed the questionnaire and answered all questions. For reasons of space limitations, we only report on the most interesting findings of our

Table 1. Mapping of questions and research questions

No	Question	RQ1	RQ2	RQ3
A1	What is the number of employees working in your organization?	✓		
A2	How many employees are working on software development in your organization?	✓		
A3	How many employees are working on your currently running project?	✓		
A4	Which of the following activities do you perform within your organization?	✓		
B1	What is the average number of users per day as anticipated at design time?	✓		
B2	What is the peak number of concurrent users during operations?	✓		
B3	Which of the following components are used in your web application?		✓	
B4	Which of these aforementioned components are obtained from a cloud service?	✓	✓	✓
B5	Suppose you start a new project or a major re-engineering of an existing application, which of these aforementioned components will you use again for Web applications that have been rebuild or evolved?		✓	
B6	What is the average lifetime of your application in number of years?			✓
B7	Within your organization how many applications share the same overall architectural design?		✓	
B8	How often do you release new features?			✓
C1	Which of the following activities are typical for software projects you have worked on?	✓	✓	
C2	Which of the following process methods are used in your projects?	✓		
C3	What types of tools do you use during your design process?	✓	✓	
C5	How do you ensure that knowledge about features, implementations, design decisions etc. is maintained?	✓		
D1	What are your three top priorities in software development?	✓	✓	✓
D2	What are your three top priorities in software development when you need to successfully maintain software in the long term?	✓	✓	✓

study. Our study database, which contains the questionnaire and all responses, can be found on http://2question.com/q1q3/.

The questionnaire has four sections. The first section (A) includes questions about the organization, role of the respondent and previous experience. The second section (B) concerns the applications developed. The third section (C) addresses the design, development and maintenance. The last section (D) concerns priorities regarding software development and software maintenance.

In the remainder of this section, we present the results and most relevant answers for every research question. Additionally we discuss results of supporting questions and control questions. This section ends with an interpretation of the results, discussion and expected and remarkable outcomes.

3.1 Analysis RQ1: How are Web and Mobile Applications Designed and How is Design Knowledge About These Applications Preserved in the Industry?

The questions most relevant to RQ1 are "What types of tools do you use during your design process?" (C3) and "How do you ensure that knowledge about features, implementations, design decisions etc. is maintained?" (C5).

In question C3, we asked participants to specify the tools[2] used for design, the time they spend on each of these tools (as a percentage of the total time spent on design activities), and the quantity of results (number of occurrences or number of produced deliverables). The design approaches, where participants spend most time on are "Experimenting, building proofs of concept" (26 %), "Documented concepts in written language like Word documents" (22 %) and "Sketches like annotated block/line diagrams" (19 %). With 11 % of total design time, technical documentation (e.g. UML, SysML, ERD, Database models) received the lowest score. In terms of quantity, the top three answers were "Verbal communication" (14), "Sketches like annotated block/line diagrams" (6) and "Experimenting, building proofs of concept" (3).

For knowledge preservation (question C5), the top three methods used in terms of spent time (percentage of overall time spent on knowledge preservation) are "Documented concepts in written language like Word documents" (26 %), "Documented code (with tools like JavaDoc, JSDoc or no tools)" (26 %) and "Verbal communication" (17 %).

Additionally, we asked participants about their top three priorities during software development (question D1) and software maintenance (D2). During *development time*, the top three priorities are "Quality" (7,2 %), "(Functional) requirements" and "time-to-market" (both 6,6 %), and "Maintainability" (4,8 %). During *maintenance*, the top three priorities are "Documentation" (18 %), "Code quality" (17 %) and both "Architecture" and "Maintainability" (6,8 %).

[2] The term *tool* is used in a wide sense here, covering among others UML, free-text, but also conversations and informal whiteboard sketches.

3.2 Questions Related to RQ2: Which Parts of the Software Architecture are Re-used Across Modern Web Applications?

The most relevant question that relates to this research question is B5: "Suppose you start a new project or a major re-engineering of an existing application, which of these aforementioned components will you use again for Web applications that have been rebuilt or evolved?"

The top three reuslts from C5 are "Webservice API (eg. RESTful, SOAP)" (86 %), "SQL Database(s)" (83 %) and "Server side web frameworks" (79 %)

A supporting question is B7: "Within your organization how many applications share the same overall architectural design?" 76 % of the applications share between 21 % and 80 % of the overall architectural design. This is equally distributed over the three categories. 21 % are from applications that share almost all components. The rest (1 %) does not share any component. Another supporting question is B3 where participants where asked about the types of software components they typically use in applications. The components mentioned most prominently (53 %) are: *Build Tools*, *Test tools*, *Server Side Frameworks*, and *Web Services*.

3.3 Questions Related to RQ3: What is the Average Life Expectancy of Modern Web Applications?

The most relevant question related to this research question is B6: "What is the average lifetime of your application in number of years?".

62 % of the applications have a lifetime between 1 and 5 years. 14 % of the applications have a lifetime of more than 10 years. The lifetime of an application determines the selection of components. For start-up companies, the initial application architecture will be sufficient for the first period. When growing in number of customers, transactions, and processes, we expect that the initial application has to be replaced with a scaleable architecture and infrastructure.

3.4 Interpretation RQ1-RQ3

In this section, we discuss the results regarding all three research questions.

In many software engineering curricula, students are taught to use (semi-) formal modeling languages like UML for designing software before coding. In contrast to this, we found that technical documents are not intensively used for design purposes in the software engineering industry. Instead, at least for mobile and web applications, the design process is primarily driven by verbal communication and informal sketches. This is in line with Sonnenburg, who describes software design as a collaborative creative activity [4], which benefits from approaches that are not constrained by fixed notations and formalisms.

On the other hand, we found that projects create more output in the shape of technical documentation than in other forms. This may be surprising at first, as less time is spent on technical documentation. On the other hand, there may be a causal relationship between those two aspects, i.e. software engineers spend

less time on technical documentation, because they are reluctant to spend time on non-engineering activities, i.e. activities that are no integral part of the built process.

In question C5, we assume a typical division between development and maintenance, in which developers in a project are not responsible for deployment and maintenance of applications. In this scenario, documentation is crucial for deployment and maintenance, as well as for managing responsibilities [5]. However, most participants chose "Verbal communication" as the primary method for handing over the code to other team members. In discussions with software engineers in the pilot group and remarks from participants, we found that engineers rather rely on proven practices in their teams, rather than formal methods, to design, develop and maintain applications. One of these proven practices is the use of verbal communication in weekly team meetings to discuss code and design. These discussions aim at improving the quality of the code by reviewing the contributions for that week and sharing the concepts and implementations.

In line with [6–8], we did not expect that webservice API's (SOAP, RESTful) would be the most re-used architectural assets (question B5). We had rather expected that data would have a higher value both for business and for software engineers and thus would be more often re-used that services.

With B6, we expected that the average life time of an application will be within 3 to 5 years (as in [9]). This is related to IT expenditures that are typically budgeted from capital expenditures. Capital expenditures have a typical amortization of 5 years. Nowadays, companies do not have to invest in costly server infrastructure anymore (capital expenditure). Instead, web and mobile applications are typically deployed in cloud environments, in which infrastructure is payed for as-a-service and is thus operational expenditure [10]. Furthermore, software engineers typically change their employer or job-role between 2 years [11] and 4.6 years [12]. Finally, software engineers typically favor building from scratch over brown-field applications that have been patched over the years. In the latter cases, the technical debt exceeds the cost of re-building from scratch.

4 Threats to Validity

In this section, we discuss possible threats to the internal and external validity of our findings. A common threat to internal validity in questionnaire-based surveys stems from poorly understood questions and a low coverage of constructs under study. The former threat was mitigated to a large extent by piloting the questionnaire with three participants form the target population. We asked these participants to fill in the questionnaire. Afterwards, they were asked to describe their interpretations of the questions and their answers. We used this input in multiple iterations to revise the questions and answering options. We addressed construct validity by explicitly mapping the questions of our questionnaire to the research questions (see Table 1) and by making sure that each research question is covered by multiple questions in the questionnaire.

External validity is concerned with the degree, to which the study results can be generalized for a larger subject population [13]. We used statistical methods to analyze whether our results are significant. Mason et al. postulate that, as a rule of thumb, questionnaires require between 30 and 150 responses in order to yield valid responses [14].

We had a total of 73 respondents; 39.7 % of whom answered all questions. Thus, we suppose that the number of respondents is sufficient.

Two remarkable outcomes from the questionnaire (questions C3 and C5) are (1) that technical documentation is less popular than plain text documentation and (2) that continuity of knowledge is achieved primarily through verbal communication. We calculated the variance and standard deviation of our responses. For C3 the variance is 0.2 and thus very low; for C5 the calculated mean is 423, the standard deviation is 193 and the weighted value for verbal communication is 425. The actual weighted value deviates by 2 points only. Thus, the results with respect to our most surprising outcomes are statistically significant.

5 Conclusions and Future Work

In this paper, we investigated how web and mobile applications are designed and documented. We found that verbal communication and informal sketches are clearly preferred over modeling languages. To preserve and transfer application-specific knowledge, companies deem code documentation equally important as technical documentation. Furthermore, for many companies, verbal communication is the primary approach for transferring knowledge within teams. This may be surprising at first as it bears the risk that knowledge gets lost, because of key employees leaving the company or a lack of communication in teams. However, web and mobile applications are primarily developed in small teams using agile development processes. Such development approaches rely heavily on verbal communication, and practices like daily stand ups in Scrum achieve that knowledge is widely spread within the development team.

Another remarkable outcome is the very high degree of architectural re-use across projects. In particular, we found that web-service APIs, SQL databases and server-side frameworks are re-used across projects in more than 80 % of the cases. This is certainly impacted by the focus of our research on web and mobile applications. Teams build up knowledge and expertise in certain technologies and exploit this knowledge to a large degree for reasons of efficiency.

Regarding the average expected life-time of web and mobile applications, we found that most applications ($\sim$ 60 %) are built for being rather short-lived (1–5 years). Further investigation is required to understand the reasons for this phenomenon.

As explained in the introduction, we will use these results for creating an architecture framework streamlined for web and mobile applications. The framework will anticipate the reluctance to produce (semi-)formal documentation and the high degree of technological re-use. We will conduct further research to understand the impact of the short life-times of such applications on the effort found reasonable for producing written documentation.

We started this research with a questionnaire to obtain quantitative data. The next phase in our research plan is to conduct interviews to collect qualitative and more in-depth data.

References

1. van Heesch, U., Avgeriou, P.: Mature architecting - a survey about the reasoning process of professional architects. In Proceedings of the Ninth Working IEEE/IFIP Conference on Software Architecture, pp. 260–269. IEEE Computer Society (2011)
2. Ciolkowski, M., Laitenberger, O., Vegas, S., Biffl, S.: Practical experiences in the design and conduct of surveys in empirical software engineering. In: Conradi, R., Wang, A.I. (eds.) Empirical Methods and Studies in Software Engineering. LNCS, vol. 2765, pp. 104–128. Springer, Heidelberg (2003). doi:10.1007/978-3-540-45143-3_7
3. Teddlie, C., Fen, Y.: Mixed methods sampling a typology with examples. J. Mixed Methods Res. **1**(1), 77–100 (2007)
4. Sonnenburg, S.: Creativity in communication: a theoretical framework for collaborative product creation. Creativity Innov. Manage. **13**(4), 254–262 (2004)
5. Carzaniga, A., Fuggetta, A., Hall, R.S., Heimbigner, D., van der Hoek, A., Wolf, A.L.: A Characterization Framework for Software Deployment Technologies. Colorado State Univ Fort Collins Dept of Computer Science (1998)
6. Teece, D.J.: Capturing value from knowledge assets: the new economy, markets for know-how, and intangible assets. Calif. Manage. Rev. **40**(3), 55–79 (1998)
7. Rayport, J.F., Sviokla, J.J.: Exploiting the virtual value chain. Harvard Bus. Rev. **73**(6), 75 (1995)
8. Howard, R.A.: Information value theory. IEEE Trans. Syst. Sci. Cybern. **2**(1), 22–26 (1966)
9. Tamai, T., Torimitsu, Y.: Software lifetime and its evolution process over generations. In: Proceerdings, Conference on Software Maintenance, pp. 63–69. IEEE (1992)
10. Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., et al.: A view of cloud computing. Commun. ACM **53**(4), 50–58 (2010)
11. Eriksson, T., Ortega, J.: The adoption of job rotation: testing the theories. Ind. Labor Relat. Rev. **59**(4), 653–666 (2006)
12. U.S. Bureau of Labor Statistics. Employee tenure summary, September 2014. http://www.bls.gov/news.release/tenure.nr0.htm. Accessed 28 Mar 2016
13. Kitchenham, B.A., Pfleeger, S.L., Pickard, L.M., Jones, P.W., Hoaglin, D.C., El Emam, K., Rosenberg, J.: Preliminary guidelines for empirical research in software engineering. IEEE Trans. Softw. Eng. **28**(8), 721–734 (2002)
14. Mason, M.: Sample size and saturation in PhD studies using qualitative interviews. Forum Qual. Sozialforschung/Forum: Qual. Soc. Res. **11**(3), Art. 8 (2010). http://nbn-resolving.de/urn:nbn:de:0114-fqs100387