A Proposal on Validation of an Agile Architecture-Modelling Process

Zoya Durdik
FZI Research Centre for Information Technology
76131 Karlsruhe, Germany

Abstract: Although considered to be incompatible, a combination of agile methods and architectural modelling could open a new perspective in software development. On the one hand, agile methods offer the flexibility required today. On the other hand, architectural modelling is considered as a pre-requisite for the systematic cross-project reuse and for the resulting increase in software developer productivity. In this paper, I define an approach for the empirical validation of a novel process for agile architectural modelling that drives requirements elicitation through the use of patterns and components. I point out the challenges connected to the process development, its design variants (degrees of freedom) and propose a validation concept.[1]

1 Introduction

While agile methods become more and more widespread in the development community [Bab09], a debate about the place of architectural modelling in agile methods is gaining an increasing attention [ABK10].

Agile methods avoid overhead, i.e., activities and artefacts that do not directly contribute to the goals of the project. This fact can be seen in a strong contrast to architectural modelling [ABK10]. This is because the effort of explicitly modelling the architecture of a software system (SA) usually does not pay off during an initial development. However, it pays off rather in the evolution phase or if a cross-project reuse occurs because of the modelled SA [Gor06]. Exactly here lie the drawbacks of agile methods: cross-project reuse of software artefacts, such as patterns, reference architectures and components are not foreseen, and product-lines are not supported. Furthermore, if the architecture is of any concern at all, it is considered to be fixed throughout the project [Bab09].

The challenge addressed in my work is how to reconcile agile methods with architectural modelling in the software development process. For this purpose a cross-gained concept of an architecture modelling process was proposed by me in [Dur10]. This process is driven by the reuse of patterns and components, and itself drives the requirement elicitation by explicitly asking questions back to the customer. It is based on the agile method Scrum [SB08] and QADA [VTT08] as a concept for architectural modelling.

Process design consists of a set of points where multiple decision options are available at a

[1] This work has been supported by the German Federal Ministry of Education and Research (BMBF), grant No. 01BS0822. The author is thankful for this support.

time. Such multiple design decisions are referred to as degrees of freedom (DoF) defining the design of a process. Decisions on what option to choose shall be taken with a strong focus on applicability of the process in practice. Therefore, one of the major challenges while designing the process is to analyse available DoF and to propose the right and well-balanced decision that can be applied on practice. Such decisions should be based on empirical studies, statistical data and/or experiments. Exactly this is usually not the case in a typical process design, where decisions are then first validated in praxis, what can lead to the expensive and time-consuming mistakes. Another challenge is a reasonable trade-off between keeping the process light-weight, keeping the software product maintainable, and making a cross-project reuse possible. Finally, the fundamental validation of such a process is a great challenge as well.

The contribution of this position paper is the discussion of a) degrees of freedom of the proposed agile architecture-driven modelling and requirement engineering (RE) process, and b) possibilities of a fundamental validation of such a process.

This paper is organised as follows: Section 2 briefly presents the overall concept of the proposed architecture-driven modelling and RE process. Section 3 describes several examples of DoF and their relation to the validation concept of the process. Related work is listed in section 4. Finally, Section 5 provides conclusions and an outlook.

2 Basic concept of the agile architecture-driven modelling an requirements engineering process

The goal of my work is to develop a software development process with agile architectural modelling where architectural modelling drives requirements elicitation through the use of patterns and components. This process shall be in line with agile principles, but support software development starting from RE and considering later maintenance and evolution phases. The cross-grained process has been proposed in [Dur10]. Due to the limited place only the brief description of the concept can be presented here.

The proposed process is iterative and incremental, and is summarized in Fig. 1. Each iteration consists of a set of phases – steps that form a single process increment (see Scrum: Sprint). These steps are: initialization, architectural design and components selection that are accomplished with a goal-oriented requirements elicitation. The process steps are repeated in iterations a) inside of each step, and b) together. Each step consists of a set of sub-steps ot shown in the figure, for more details please refer to the above reference.

During these iterations, the artefacts are created according to the demand of the current process stage, therefore, keeping the process lightweight. The innovation lies in the fact that architectural design and the potential reuse of component candidates are drivers of the requirements elicitation process. Please note that the system architectural design is not intended to be complete after the iteration, only a part of the system is designed during it.

Architectural design and component selection shall be supported correspondingly by a pattern catalogue and a component repository. Such catalogues contain a detailed description,

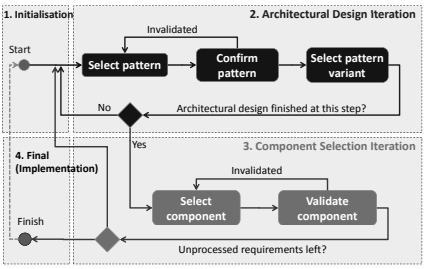

Figure 1: A simplified abstraction of the agile architecture-driven modelling and RE process

i.e. a goal, benefits, drawbacks, application cases, as well as connected or similar patterns or components. It also contains a set of questions that shall help to confirm or invalidate the decision on a pattern or a component and its variant. These questions may help to elicit additional requirements, if the initial information on the system was not sufficient.

This process, however, shall be seen as an example, as its single steps are targets of current and future research. Identified DoF for this steps, i.e. amount of architecture modelling is required, are reviewed in Section 3.

In addition, the benefits of SA need to be exploited in order to support an overall agility. This includes the use of SA for cost estimation, early evaluations of architectural design decisions, and the use of SAs to elicit the right requirements.

3 Degrees of freedom, and concept of the process validation

This section discusses the validation of the proposed process, and exposes its connection to the identified degrees of freedom that are described in detail later on.

Any process usually consists of several steps, and any of these steps can have multiple implementation and execution alternatives. In this paper, such alternatives are referred to as “degrees of freedom” (DoF) of a process design. In the majority of the existing processes, the decisions which variant of a DoF to choose were taken either randomly or based on a personal feeling or the experience of the designer. As the validation of a process is generally a very complicated and labour-intensive task, the possible solution here is to construct a process as a set of “building blocks”, so that each of such blocks can be validated individually. The decisions on each design step and for each DoF shall be taken based on the results of empirical studies, experiments and/or statistics.

Therefore, the refinement of the here proposed process will be executed as a set of smaller steps – building blocks, each explicitly validated. These steps also include evaluation of possible DoF, and validation of design decisions taken in this matter. This will serve as a solid ground for the validation of the complete process assuring the validity of its steps. The validation of a complete process includes validation of the hypothesis that refactoring overhead at the later life-circle stages of a product will be reduced by the explicit architecture modelling supported from the beginning. The validation will involve

2 students groups participating in a controlled experiment. A crossover between the groups shall reduce influence of the learning effect on the experiment results. Nevertheless, a complete and fundamental validation of the process might still not be possible in the frame of my work due to the scale of resources required for it. DoF, which are already identified, will serve as building blocks for the process design. They are listed in the following. Please note that only few DoF could be described here because of the limited space.

DoF 1. How much of the architectural design and architectural modelling is required. Usually the amount of architectural modelling required in a certain project (respectively, project's stage) is determined by the personal experience of the person responsible for the system design (e.g. chief architect or the Team in agile methods). As this research question is not new, and there can be no universal solution, my goal is to analyse currently available solutions and to propose a set of indicators helping to support such decisions.

DoF 2. Decision if the selection of design patterns shall take place before component selection or it shall be done vice versa. Here we have three possible process design decisions. The first option is to introduce the architectural design first, as supposed by the first version of the proposed process. However, the second option is to introduce the component selection first. In this case the design will be built upon the available components. Finally, the third option is to let the person executing the process decide on what is most suitable for the current project. In this case it is important to provide guidelines supporting the selection of the most suitable variant. Such guidelines can be created based on the analysis of a) the statistics on the availability of reusable components (component repositories) in practice, and b) results of an empirical study comparing these two ways.

DoF 3. The fact if architectural decisions are taken at the beginning of the process or are postponed depends on the statistics about to which extend the SA changes during a system's evolution process. If the first SA draft is maintained throughout the development and evolution phases, then a basic SA concept can be defined already during the initial phase of the process. Otherwise, architectural decisions will have to be postponed. For how long such decisions can be postponed is an open research question, as there is no empirical data available to answer it at the moment.

Therefore, an empirical study dedicated to these topic is currently planned. The challenge here lies in the fact that there is no explicit architectural data available for the analysis. Architecture models of open source projects are usually not maintained (or even not available at all). In order to overcome this challenge a reverse engineering tool SoMoX (http://www.somox.org) will be applied to recover the missing models from version to version. So it will be possible to trace if some new features added in the newer versions have produced a significant impact on the SA. It has to be also assured that the changes identified by SoMoX were really introduced into the initial SA plan through new requirements, and are not resulting from the reverse engineering partitioning for components.

Additionally, the results of the study might strongly depend on the domain. This causes another difficulty, as most of the open source projects available for the analysis more or less belong to the category of technical environment applications (e.g. Eclipse or Apache).

Some further examples of the DoF are: decision if SA design and system implementation steps shall be organized as Sprints in Scrum, and a definition of the size of these

steps; decision on form of the SA artefact for the process; and decision if the requirements elicitation shall be frozen during the development phase.

4 Related Work

Generally, most of the articles appearing in this area tend to be rather introductory than practical, and do not propose any exact solution of the problem described above. Several relevant practical approaches are presented in this section.

So [Amb02] provides an introduction into agile methods, UML-modelling and describes general agile modelling methodologies. However the advice given is general, mostly concentrating on the "best practices" of Extreme Programming (XP), and concerned with a very general management level.

The approach presented in [ZBJ04] proposes improving the SA evolution process by mining architecture and design patterns. Although the approach uses a similar idea of information extraction out of patters, its goal is to support the system's evolution and to extract general scenarios for it.

A more practical approach is presented in [Mad10]. It is based on Scrum, XP and sequential project management. Its main goal is SA for communication, quality attributes and technology stack establishment purposes. It does not affect the RE process. Design patterns are used to give a form to implementation, but not to support requirement elicitation and further architectural design. Besides that, the approach seems to be aimed for the similar project types (insurance software systems), as it uses former architectural decisions.

A closely related article [Nus01] states that a SA provides a basis for further requirement discovery and determination of the alternative design solutions. The proposed method is an adoption of the spiral life-cycle model and the author describes it as a complementary to XP. However, the requirements here emerge from: a) the view on models and prototypes b) from the commercially available software (COTS, commercial off-the-shelf software), which shall narrow the possible architectural decisions. The article mentions only the reuse of COTS and requirement identification connected with them. Design patterns are mentioned for "fitting a component-based development approach into the development process". It is a high level article, it neither proposes an exact way how the information from the analysis can flow back from patterns to requirements, nor gives it exact details about the method itself.

Additionally, there is a set of studies on influence of existing SA on the RE, i.e. [MFM09] or [FSH+10], that will be used for a "case study 3" design (Section 3, DoF 3).

5 Conclusions

In this paper, I have discussed a way to validate a novel architecture-driven modelling and RE process proposed by me in [Dur10]. I have listed the major challenges connected to

the process design. I have also described several DoF of such process, pointing out their role in the validation of the whole process.

The future work will have to concentrate on the research of the here presented DoF. According to their type, a required empirical study or an experiment will be designed and conducted to support the selection of the step alternatives of the process, thus assuring that the process steps are based on the experimental / empirical data. In the ideal case, every decision on the process alternative will be validated. This will in no case replace the validation of the proposed process as a whole, but will contribute to the confirmation of the process viability, as well as serve as an additional delta to the already existing processes.

References

[ABK10] P. Abrahamsson, M.A. Babar, and P. Kruchten. Agility and Architecture: Can They Coexist? *Software, IEEE*, 27 , Issue 2:16 – 22, 2010.

[Amb02] S. Ambler. *Agile Modeling: Effective Practices for eXtreme Programming and the Unified Process*. John Wiley & Sons, 2002.

[Bab09] M. A. Babar. An exploratory study of architectural practices and challenges in using agile software development approaches. *Joint Working IEEE/IFIP Conference on Software Architecture 2009 & European Conference on Software Architecture*, pages 81 – 90, 2009.

[Dur10] Z. Durdik. Architectural Modeling in Agile Methods. *Fifteenth International Workshop on Component-Oriented Programming 2010 at CompArch 2010*, 2010-14:23 –30, 2010. Awarded with CompArch Young Investigator Award.

[FSH+10] R. Ferrari, O. Sudmann, C. Henke, J. Geisler, W. Schafer, and N. H. Madhavji. Requirements and Systems Architecture Interaction in a Prototypical Project: Emerging Results. *16th International Working Conference on Requirements Engineering: Foundation for Software Quality, Lecture Notes in Computer Science*, Volume 6182/2010:23 – 29, 2010.

[Gor06] I. Gorton. *Essential Software Architecture*. Springer, 1 Edition, 2006.

[Mad10] J. Madison. Agile-Architecture Interactions. *Software, IEEE*, 27 , Issue 2:41 – 48, 2010.

[MFM09] J. A. Miller, R. Ferrari, and N. H. Madhavji. Characteristics of New Requirements in the Presence or Absence of an Existing System Architecture. *2009 17th IEEE International Requirements Engineering Conference*, pages 5 – 14, 2009.

[Nus01] B. Nuseibeh. Weaving Together Requirements and Architectures. *Computer*, 34 , Issue 3:115 – 119, 2001.

[SB08] K. Schwaber and M. Beedle. *Agile Software Development with Scrum*. Pearson Studium, 2008.

[VTT08] Technical Research Centre of Finland VTT. QADA, Quality-driven Architecture Design and quality Analysis. http://virtual.vtt.fi/virtual/proj1/projects/qada/, 2008.

[ZBJ04] L. Zhu, M. A Babar, and R. Jeffery. Mining patterns to support software architecture evaluation. *Fourth Working IEEE/IFIP Conference on the Software Architecture, 2004*, pages 25 – 34, 2004.