An Architecture-centric Approach for Goal-driven Requirements Elicitation

Zoya Durdik
Research Center for
Information Technology (FZI)
76131 Karlsruhe, Germany
durdik@fzi.de

ABSTRACT

Software system development typically starts from a requirement specification followed by stepwise refinement of available requirements while transferring them into the system architecture. However, the granularity and the amount of requirements to be elicited for a successful architectural design are not well understood. This paper proposes a process concept to support system development with the help of an architecture-centric approach for goal-driven requirements elicitation. The process focuses on multiple quality dimensions, such as performance, reliability and scalability, and at the same time shall reduce costs and risks through early decision evaluation. The main contribution of this paper is a novel process where not only requirements can drive architectural design, but also architectural design can selectively drive requirement elicitation with the help of hypotheses connected to the selected architectural solutions. The paper concludes with a discussion on its possible empirical validation.

Categories and Subject Descriptors

D.2.1 [**Software Engineering**]: Requirements/Specifications; D.2.11 [**Software Engineering**]: Software Architectures; D.2.9 [**Software Engineering**]: Management

General Terms

Design, Documentation

Keywords

Requirements elicitation, software architecture, architectural modelling, architectural solutions, design patterns, architectural decisions, development process

1. PROBLEM STATEMENT

Because of the high complexity, strict time and resource constraints, the derivation of a complete requirement specification before proceeding to the architectural design is probably impossible in practice [10]. This fact is especially common for non-functional requirements, but also includes high-priority functional requirements [21]. Moreover, it is not clear how many requirements and shall be elicited and at which level of detail before the process can proceed to the architectural design. An architect usually starts the software design process with such incomplete requirements specification. He transforms available requirements stepwise into elements of the architecture, which in this case are drivers of architectural design. Even already available requirements can be misinterpreted or be inconsistent [10]. The later such inconsistencies and new important requirements are discovered, the higher is the cost and overhead connected to the required system refactorings and implementation [21]. In this case, success of the design process strongly depends on architects experience, quality of the initial requirements and complexity of the system to be developed.

However, not only requirements can drive architectural design, but also architectural design can drive requirement elicitation. Given a catalogue of architectural solutions (architectural styles, patterns, components, services, domain specific solutions, etc.), the solutions could be annotated with sets of questions unique for each solution. Questions would serve for the identification of a solution satisfying a subset of selected requirements, and for the validation of solution suitability and applicability at architectural design. These specific questions sets are referred to as hypotheses in this paper. Answering the hypotheses would contribute to implementation of the available requirements (and trace link creation), and to the elicitation of the new requirements (which are then captured and traced). The advantages would be the following: faster recognition of new relevant requirements, right level of granularity, early detection of contradictions between requirements, and less time spent on at the moment unnecessary requirement elicitation.

In this paper I propose a process concept to support system development with the help of an architecture-centric approach for goal-driven requirements elicitation. The process focuses on multiple quality dimensions, such as sustainability, performance, reliability, security and scalability, and at the same time shall reduce costs and risks through early decision evaluation. Explicit architecture modelling drives selective requirement elicitation through the hypotheses connected to the selected solutions.

This paper is organized as follows: Section 2 presents research objectives and Section 3 the proposed process; Section 4 describes research status. Section 5 provides an overview of related work, and Section 6 concludes.

2. RESEARCH OBJECTIVES

The goal is to improve the requirements elicitation phase by establishing a stronger connection between architecture decisions and requirements, and to support system development with focus on multiple quality dimensions, such as sustainability, performance, reliability, security and scalability, reducing costs and risks.

In order to achieve this goal I propose a novel architecture-centric tool-supported software development process described in Section 3. The expected contributions of the process are the following:

- active driving of selective requirements elicitation at the appropriate level of granularity, and early detection of contradictions between requirements through the use of available solutions in the architecture solutions repository;
- reduced risks and costs through early decision evaluation and reuse of architectural knowledge;
- explicit focus on multiple quality dimensions through reuse of specific solutions (e.g. patterns targeting scalability or performance)
- support explicit architecture modelling, documentation of system design and design rationale building up trace links between solutions, decisions, requirements and architecture model elements.

I argue that the approach is beneficial for domains where explicit modelled architecture design plays a significant role, such as business information systems domain (e.g. SAP system).

3. PROPOSED APPROACH

The process consists of iterations of the three major phases: initialisation, architectural solutions selection (architectural styles and patterns, existing services and components, domain specific solutions, etc.) and design implementation, as illustrated on Fig. 1. Selection and implementation phases are themselves iterative and incremental (both alone and in combination), and shall be executed as Scrum Sprints [20]. In the following the process foundations, and major phases are described in detail.

Foundations. The process builds upon Scrum, extendible and relatively lightweight agile method of software development [20], and a process proposed by me in [3], [4], and summarized in [5] (this earlier proposed process was targeting agile architecture modelling). For definition and modelling of software architectures I build upon the Palladio Component Modelling approach (PCM) that is a component-based architecture model approach and tool-chain [1]. PCM has five views on software architecture: repository, assembly, behaviour, deployment and usage. It is implemented as a set of Eclipse plug-ins for prediction of non-functional properties (performance and reliability) at the architectural level, thus enabling early architecture decision validation. It will be additionally enhanced with a requirements view for requirements capture and management, and a trace links mechanism. The decomposition of the problem domain is based on [22] and the extended problem frames approach [17, 2]. The idea to extend focus from reliability and performance to multiple other quality dimensions was inspired by [18].

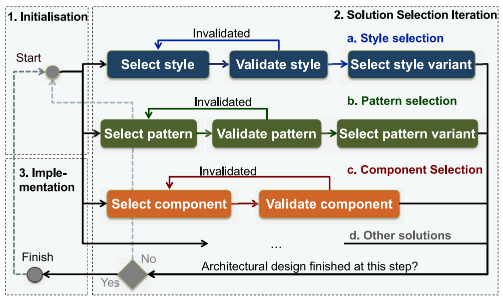

Figure 1: Collaboration between process phases

Initialisation. The process does not consider initial requirements elicitation, assuming it could be successfully executed before start. The initially elicited requirements shall be recorded and prioritised conforming to the special requirements description template and requirements meta-model (both currently in development) and shall be managed in PCM requirements view. The template is the enhancement of Scrum Product Backlog [20] and Architecture Backlog, developed in the [19] under my supervision. After initialisation work the architect can proceed to the solution selection. In later process iterations during initialisation phase the architect goes on with the refinement of the next most important requirements that are not yet implemented and described with the help of the template.

Solution selection. To support requirements refinement into architecture, I propose a rule-based expert system containing a repository of architectural solutions (styles, architecture-relevant patterns, existing components, services, domain specific solutions, etc.). Each of the solutions is stored in the solution repository together with the detailed description of its goal, context, instantiation details and hypotheses. *Hypothesis* is a set of questions specific for a solution, aiming not only to discern between similar solutions, but also to validate the suitability of the selected solution. Solutions are annotated with information about their relevance to the quality dimensions (e.g. improves security, reduces performance).

While there is a rather limited number of architectural styles, the number of e.g. architecture relevant patterns or services can be next to unlimited. The system has to support several solution selection mechanisms to scale and to propose a limited number of only the most relevant solutions.

First, the initial set of available solutions is reduced through the selection of a category (e.g. pattern, style, component, categories within patterns), predefined keywords and the most relevant and prioritized quality dimensions. Solution annotations support the focus on the desired quality dimensions. The non-functional properties of the designed system can be later evaluated with the help of PCM, as solution annotations are not sufficient to make conclusions on the overall expected system quality.

Second, the architect will be able either to browse through the reduced choice of solutions, or answer special questions narrowing the choice towards one possible solution group, and then a particular solution in the group. These questions target the difference between groups of solutions and

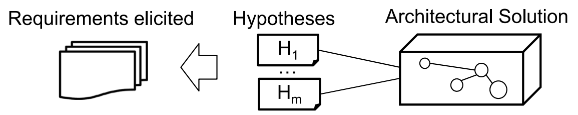

Figure 2: Requirements elicitation from hypotheses

solutions within these groups solving the same problem. For example, concrete table inheritance, single table inheritance and class table inheritance patterns solve the same problem of mapping from objects to relational database tables as they do not support inheritance [7]. Such patterns would belong to one group in the repository. To discern between them in the group, more specific questions contained in the connected hypotheses are required (e.g. questions concerning performance or database technology preference).

Moreover, answering questions contained in the hypotheses can help to elicit new requirements that were otherwise not evident, as shown in Fig. 2. Coming back to the example: it might become clear that due to the performance issues in the database, the access load shall be spread between different tables (concrete table inheritance), or that the database does not have an optimisation and any loss of space shall be avoided (single table inheritance). In this case a new requirement is elicited that otherwise might have remained undiscovered till the point where consideration of it would have been highly expensive. This example requirement is non-functional, but the approach also supports elicitation of functional requirements.

When the architectural style is selected, one goes on analysing requirements having the highest priority and implementing them in the design, refining the architecture and also architecture model. An expert system that proposes suitable solutions shall reduce architecture design overhead at the same time stimulating explicit architecture modelling and its further refinement. Ideally, modelled architectural solutions could be imported directly into PCM. An architecture model created in PCM can than be simulated to predict e.g. the system's performance or reliability. This prevents unsuitable architecture solutions from being implemented and, therefore, reduces risks and makes development more agile. Thus, the focus on quality dimensions is supported by the solution quality annotations at the one side, and PCM at the other side.

Additionally, using hypotheses to validate the solution enables traces between architecture elements, requirements and decisions, including their rationale. The rationale contains information on requirements that were crucial for the selection of one particular solution. An advantage is that requirements that are crucial for the making of a concrete decision, and therefore are the most relevant for certain architectural elements, can be filtered and marked automatically. This helps to avoid having a forest of trace links themselves requiring a lot of maintenance effort (e.g. when architecture needs to be refactored).

Implementation and next iterations. A design draft (partial or complete) created within architecture design iterations shall be presented and explained to the implementation team that can start implementing identified tasks with the highest priority (Scrum). Architect goes on with the architectural design iterations refining architecture. This paper does not touch upon the issue of implementation itself, assuming that design decisions can be successfully broken down into tasks, and then implemented in Sprints.

4. RESEARCH STATUS

In [3] I initially proposed a process targeting agile architecture modelling, and in [4] its validation concept. These proposals are summarized in [5] (to appear). Current work in progress incorporates: 1) development of the requirements description template and a corresponding meta-model, and 2) development of the expert system prototype. The prototype containing an initial set of solutions (only architectural styles in the first version) shall be finished till the end of 2011. The hypotheses for the prototype base on the literature and existing patterns catalogues (e.g. [7, 9]). The preliminary validation is planned for the beginning of 2012.

This work is supervised by Prof. Dr. Ralf H. Reussner, Karlsruhe Institute of Technology.

5. RELATED WORK

The process proposed in this abstract supposes decisions to be captured through the hypotheses and questions connected to them. It differentiates it from majority of approaches dealing with the design decisions and their capture, e.g. [24, 11]. A Goal Solution Scheme [18] maps quality goals and goal refinements to architectural principles and solutions, and can be used complementary to my approach. It focuses on non-functional requirements and architectural solutions, similar to the ones proposed in this paper, but does not consider requirements elicitation, and early decision evaluation. The main drivers of the approach are quality goals that serve as weights when proposing a solution. These arguments are also valid for the [10, 9, 12, 8] and [15], the latter evaluates multiple available design decisions basing on selected quality attributes.

Studies in [14, 13, 6] confirmed influence of existing software architecture on the requirements engineering, however they do not consider using this influence for the requirement elicitation. The idea to use architecture as a basis for further requirement discovery and determination of the alternative design solutions is presented in [16]. However, authors propose to use the view on models and prototypes and commercially available software to emerge new requirements. Design patterns are a way of implementation. The authors give very few details about the method itself. To my best knowledge, the author has no further publications directly to this topic. Another method using similar idea is [23]. It uses information extraction to improve the architecture evolution process by mining architecture and design patterns, however, its goal is to support the system's evolution and to extract general scenarios for it.

6. CONCLUSION

I argue that architectural design can be a driver for a goal-oriented requirement elicitation. First, an expert system containing various solutions shall support architect transforming requirements into architecture. Second, architectural solutions themselves contribute to goal-oriented requirement elicitation through unique hypotheses connected to each solution in the repository. Validation or invalidation of this hypotheses leads to refinement or implementation of existing requirements or to the elicitation of new ones. Thus, not only requirements lead to architectural decisions, but also architectural decisions lead to new requirements that are may be crucial for the system sustainability and would be otherwise discovered much later.

A solution repository supports architectural knowledge reuse through use of patterns, styles, services, etc. contained in a solution repository. No expert knowledge of possible architectural solutions is required by a person using the expert system, as he is supported by the expert system, which can be also used as a knowledge repository itself. Selection of solutions from the repository leads to the creation and continues refinement of architectural models. These models can then be evaluated in the PCM for the violation of non-functional requirements, such as performance and reliability.

Although not all architectural decisions can be derived from available architectural solutions and not all requirements are architecture relevant, the semi-automated proposal and integration of solutions can improve overall system quality targeting sustainability, design speed, and assure that design decisions are validated before implementation. The proposed Backlog modification will enable dealing with and management of requirements that are irrelevant to the architecture. Additionally, I plan to include some non-architectural best practices into the solution repository, that are important to support the selected quality dimensions .

The cost of efficiency benefits of software processes are hard to empirically demonstrate by nature. Therefore, I plan to validate that (a) the process leads to less wasted requirements, (b) more precise formulated requirements (c) traceability between requirements and implementing architectures through series of planned case studies. Of course, controlled empirical experiments would have a higher validity, but are practically hard to perform due to the high effort for the participants. Therefore, I plan to perform studies, where participants (students and researchers colleagues) have to do only specific subtasks in existing documents, such as finding ambiguities or lacking requirements. From a comparison of this process with a process without architectural feedback on requirements elicitation, I plan to extrapolate from finding on such subtasks to the general effect of the process. Additionally, the sufficiency of scalability mechanism will to be tested on the larger set of requirements.

I am thankful for the support of the German Federal Ministry of Education and Research (BMBF), grant No.01BS0822.

7. REFERENCES

[1] S. Becker, H. Koziolek, and R. Reussner. The palladio component model for model-driven performance prediction. *Journal of Systems and Software*, 82:3–22, 2009.

[2] C. Choppy and M. Hatebur, D.and Heisel. Architectural patterns for problem frames. *Software, IEE*, 152:198 – 208, 2005.

[3] Z. Durdik. Architectural modeling in agile methods. volume 2010-14 of *Interne Berichte*, pages 23–30, Karlsruhe, Germany, June 2010. Awarded with CompArch Young Investigator Award.

[4] Z. Durdik. A Proposal on Validation of an Agile Architecture-Modelling Process. In *Proceedings of Software Engineering 2011, Dokt.-Symposium*, 2011.

[5] Z. Durdik. Towards a process for architectural modelling in agile methods. In *7th Int. Conf. on the Quality of Software Architectures (QoSA'11)*, 2011.

[6] 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 Int. Working Conf. on Requirements Engineering*, 6182/2010:23 – 29, 2010.

[7] M. Fowler. *Patterns of Enterprise Application Architecture*. Addison-Wesley Professional, 2005.

[8] M. Galster, A. Eberlein, and Moussavi. Systematic selection of software architecture styles. *Software, IET*, 4 Issue:5:349 – 360, 2010.

[9] H. Garbe, C. Janssen, C. Moebus, H. Seebold, and H. de Vries. KARaCAs: Knowledge Acquisition with Repertory Grids and Formal Concept Analysis for Dialog System Construction. In *Managing Knowledge in a World of Networks*. Springer, 2006.

[10] P. Grünbacher, A. Egyed, E. Egyed, and N. Medvidovic. Reconciling software requirements and architectures with intermediate models. In *Software and Systems Modeling*, pages 202–211. Springer, 2003.

[11] L. Lee and P. Kruchten. Customizing the capture of software architectural design decisions. In *Canadian Conf. on Electrical and Computer Engineering*, 2008.

[12] J. Madison. Agile-architecture interactions. *Software, IEEE*, 27 , Issue 2:41 – 48, 2010.

[13] J. Miller, R. Ferrari, and N. Madhavji. Architectural effects on requirements decisions: An exploratory study. *Seventh Working IEEE/IFIP Conf. on Software Architecture*, pages 231 – 240, 2008.

[14] J. A. Miller, R. Ferrari, and N. H. Madhavji. Characteristics of new requirements in the presence or absence of an existing system architecture. *17th Int. Requirements Engineering Conf.*, pages 5 – 14, 2009.

[15] M. Nowak, C. Pautasso, and O. Zimmermann. Architectural decision modeling with reuse: challenges and opportunities. In *Proceedings of the 2010 ICSE Workshop on Sharing and Reusing Architectural Knowledge*, SHARK '10, pages 13–20. ACM, 2010.

[16] B. Nuseibeh. Weaving together requirements and architectures. *Computer*, 34 , Issue 3:115 – 119, 2001.

[17] L. Rapanotti, J. G. Hall, M. Jackson, and B. Nuseibeh. Architecture-driven problem decomposition. *In Proceedings of the 12th IEEE Int. Requirements Engineering Conf.*, pages 80 – 89, 2004.

[18] M. Riebisch. Problem-Solution Mapping for Evolution Support of Software Architectural Design. In *Proceedings of Software Engineering 2011*, 2011.

[19] F. Schad. Enhancement of an agile process model through a software architecture artefact. Student research thesis, KIT, 2010.

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

[21] I. Sommerville. *Software Engineering*. Addison Wesley, 2006.

[22] W. Wang and J. E. Burge. Using rationale to support pattern-based architectural design. In *Proceedings of the 2010 ICSE Workshop on Sharing and Reusing Architectural Knowledge*, SHARK '10, pages 1–8, 2010.

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

[24] O. Zimmermann. Architectural decisions as reusable design assets. *Software, IEEE*, 28 Issue 1:64 – 69, 2010.