Position Paper: From Enterprise Architectures to Software Architectures using Requirements Engineering

Matthias Galster  
Department of Electrical and Computer Engineering  
University of Calgary  
Calgary, AB, Canada  
mgalster@ucalgary.ca

Armin Eberlein  
Department of Computer Engineering  
American University of Sharjah  
Sharjah, United Arab Emirates  
eberlein@ucalgary.ca

Mahmood Moussavi  
Department of Electrical and Computer Engineering  
University of Calgary  
Calgary, AB, Canada  
moussam@ucalgary.ca

Abstract - Enterprise architectures represent business objectives that can be extracted during requirements engineering. After gathering these objectives in form of requirements the resulting specifications must be translated into software architectures for later implementation. This transition has proven to be a non-trivial task. Even though requirements engineering and software architectures are well established areas in the software engineering domain we still lack fundamental guidelines and rules for the bridging between these two stages. This paper presents our research position by proposing a flexible and adaptive process framework to relate software requirements based on enterprise architectures and software architectures.

Keywords: Requirements Engineering, Enterprise Architectures, Software Architectures, Transition.

1 Introduction

Enterprise architectures organize the business objectives and resources of a company to optimize the benefit of an organization. They exist independent of any software implementation and describe processes, information systems and organizational sub-units that support the main goals and strategic directions of an organization. During Requirements Engineering (RE) we discover the purpose and intention of a future software system by eliciting goals, tasks and features, often based on the enterprise architecture. Software architectures on the other hand are based on requirements and describe components, relations, and constraints of a software system under development. Researchers have made significant progress in RE as well as in software architectures, however, only in separation. The relation between requirements and software architectures still lacks a structured foundation. The importance of this relationship must not be underestimated as requirements and software architectures have tremendous impact on the remaining software life cycle.

Our motivation for research into relating requirements and software architectures is driven by the idea that progress in RE research and the effort spent on it in practice is futile when we fail to manage well the communication and translation of all requirements and related intents throughout the development process.

Section 2 of this paper gives a more detailed introduction to the problem that is put into an overall context in section 3. Section 4 shows previous attempts to bridge the gap between requirements and software architectures. A transition framework is presented in section 5. Section 6 concludes the paper.

2 Motivation

To create software support for processes and businesses of organizations we need to translate enterprise architectures into software architectures. RE can be used to capture software-related intents from enterprise architectures and describe them in specifications for their software implementation. This can be done using existing RE techniques and process models. However, the next step towards software architectures is not as clear. Currently, many organizations follow an ad-hoc, unstructured and informal procedure to represent the software requirements in software architectures. There are several reasons why it is so important to have well-defined software architectures: Architectures provide us with the ability for communication, extended analysis, and a foundation for later implementation. Poor or even non-existing software architectures tend to result in poor software products.

At the moment the transition from software requirements to software architectures mainly relies on the ability of developers, their domain knowledge, experience and intuition. This limits the traceability of design decisions and creates dependencies on the developing skills of individuals. A process-oriented transition framework would help increase the quality of development processes and software products. Nevertheless, we have to be aware that architecting will always be creative work that only can be supported by a process framework.
3 Enterprise Architectures in the Context of RE and Software Architectures

Enterprise architectures address strategic entities, functional aspects and processes that support a business of an organization. They often focus on IT aspects. An enterprise architecture should represent what exists and design what should exist but neglect physical constraints and concrete system design. Enterprise architectures can be considered as the input for software architectures because software specifications are created based on them.

Figure 1. RE as intermediate layer between Enterprise- and Software Architectures

Implementing an enterprise architecture generally starts with documenting the organization’s strategy and other necessary details. The process then documents single core aspects, business processes, and how the organization interacts with itself and external parties. Following this, RE can be seen as an intermediate step from enterprise architectures to software architectures. This becomes clear when comparing common goals of enterprise architectures and RE: understand and define business goals, operational goals and IT system goals of an organization.

As we can see in figure 1, the step from enterprise architectures to software requirements is done using RE techniques. On the contrary, the following transition from requirements to software architectures is not clearly defined. That shows the need for a guided transition for this step from enterprise architectures to software architectures as their software representation.

4 Related Work

Current research to bridge the gap between requirements and software architectures mainly extends existing RE techniques towards architecture building. The goal-based approach [7] provides guidance for the transition to meet all (non-) functional requirements. It supports the analysis of the global impact of requirements, recursive refinement, and creation of different architectural views. An extension towards agent-oriented development can be found in [8]. Perry’s work focuses on empirical foundations [14], Chung et al. provide the NFR framework [3], the Preskriptor defines a high-level architecture description paradigm [2]. Architecture Description Languages (ADLs) provide a formal description of information, components, relations and constraints [12]. However, ADLs neglect the complex modeling of non-functional aspects. Use Case Maps (UCMs) offer a lightweight notation to visually describe relationships between responsibilities of use cases [1]. They scale well and support early architectural descriptions represented as scenarios with visual behavior structures. Problem Frames structure the problem domain [4] which is similar to architecting requirements [9]. They relate requirements, domain properties and machine descriptions. A decision framework was proposed in [10] for partially automated reasoning. It supports mapping, conversion and analysis using decision trees. Direct object-oriented transition seems to be possible by transferring results of object-oriented analysis into object-oriented design. Nevertheless, this causes problems [6] because design describes internal aspects whereas analysis describes system structures from the user’s point of view. Model bridging tries to bridge requirements models and architecture models [11] to overcome the gap between high-level requirements and low-level architectural artifacts. The bridging relies on transformation and comparison of model information. Weaving requirements and architecture processes was proposed by Nuseibeh with his Twin Peaks Model [13]. The goal is to get an early understanding of the system and start construction.

Existing methodologies still leave open issues. Even if some approaches propose a direct mapping we think that this will likely not be possible in the near future. Direct mapping would require software components to be paid more attention already in the analysis and requirements phase. Moreover, all current approaches require significant human input without adequate tool support. Another important aspect is the classification of requirements and architectural aspects with respect to their impact on the architecture which has been addressed insufficiently so far. Finally, these approaches still rely on intuition and experience.

5 Generic Process Framework

The following process framework supports the transition from requirements to software architectures that is flexible and allows learning – two major features each process model should support (figure 2). Such a framework could be considered as the missing piece between enterprise architectures and software architectures.

5.1 Multi-Disciplinary Approach

As mentioned before, the transition from requirements to software architectures requires cognitive capabilities,
like creativity, synthesis and problem solving abilities. Moreover, it is essential to understand and sometimes even necessary to manipulate the information flow in the architecture process. A multi-disciplinary approach could help as it allows the inclusion of communication support, formal representations, and visual modeling techniques to leverage human understanding.

5.2 Process Definition

The role of enterprise architectures in the framework is twofold: Firstly, they provide us with requirements on which we base the software architecture. Secondly, they allow us to directly derive high level software architecture artifacts.

One integral part of the framework is the multi-aspect classification of requirements according to several criteria: a) Impact: Strength of impact and the impacted architectural aspects b) Novelty: Differentiation between known and new requirements. This allows performing a knowledge-library-based transition for known requirements and a base transition for new requirements.

The merger process requires the integration of the two solution spaces and conflict resolution between them. Artifacts might be added from one solution space, or just merged. Knowledge from this procedure will also be added to the knowledge library.

5.3 Basic Transition

The basic transition requires most effort. As we can see in figure 2 the basic transition can be separated into 3 different parts, each focusing on different aspects (RE, architecture, or transition). Based on this we can derive additional methods and tools for the basic transition. That is another factor which increases flexibility of the process model and helps when developing plug-ins (see below).

By splitting up a requirements set as shown in figure 3 we get first architectural artifacts which are either a static component, a dynamic behavior, or an interaction. Artifacts from enterprise architectures could be used to develop a skeleton in form of a high level structure, i.e., enterprise architecture artifacts might become software architecture artifacts.

5.4 Plug-ins and Hooks

In practice, the number of views and layers depends on project properties, available resources, and objectives. Therefore we propose the inclusion of plug-ins for the creation of additional views. A plug-in has to comply with a certain format. This format consists of a name = \{<text>\}, the location = \{Hn\} at which the plug-in will be integrated, and the description = \{<text>\}. The description consists of details about when and how to apply the plug-in. To create a plug-in we use differences between factors,
e.g. components, attributes, level of granularity, styles, domain properties or patterns. A sample plug-in is shown in figure 4.

<table>
<thead>
<tr>
<th>Name</th>
<th>Data flow view</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location</td>
<td>H2</td>
</tr>
<tr>
<td>Description</td>
<td>Creates architectural view that focuses on the data flow between architectural artifacts. Data flow modeling is part of RE why it is possible to focus on DFD diagrams in the RE phase and integrate them into an architecture view.</td>
</tr>
</tbody>
</table>

Figure 4. Sample plug-in

Plug-ins ensure adaptability of the process model. As the transition process is always iterative, adding a plug-in may also increase the number of iterations.

In figure 2, we see the hooks used to integrate additional plug-ins. Based on the previously mentioned 3 different parts in the transition procedure they can be used for the following manipulations when creating plug-ins:

H1: Manipulation of novelty evaluation
H2: Manipulation of RE-focused activities
H3: Manipulation of library inclusion
H4: Manipulation of integration methodology
H5: Manipulation of architectural properties

5.5 Knowledge Library

The knowledge library is a core artifact in the framework. Knowledge representation and knowledge extraction play a crucial role. The format of an entry in the knowledge library has 3 different abstraction layers. A natural language description is presented to the user. By using a structured description in XML the natural language representation is mapped to the architectural knowledge. Architectural knowledge can be of different kind, e.g. guidelines, actual components, relations, or implied risks. The knowledge extraction can be understood as the gathering of transition knowledge during the transition process. As architecting is a creative human activity the gathering is also mainly a human process.

6 Conclusion and Future Work

In this position paper we presented our current work which focuses on the transition from requirements to software architectures. Therefore, it provides a structured way when transforming enterprise architectures into software architectures. We argued that the main problem is the step from enterprise architecture-based requirements to the actual software architecture. Our overall feeling is that a transition from requirements to software architectures requires more work than extending existing RE techniques towards architecting. The introduced process framework provides flexibility by including hooks for plug-ins and learning abilities through an integrated knowledge library.

The focus of our future work is the definition of proper guidelines for the basic transition. We plan to weave requirements guidelines and architecture steps. We might consider architectural aspects already during RE.

We also plan to include plug-ins that extend the input and output options, e.g. allow the creation of UML class diagrams based on a view. Additionally, we want to find answers for the following questions:

- How do we deal with changing requirements?
- Is it possible to split a set of requirements? How do we deal with interrelated requirements?
- How can we evaluate if an architectural solution is suitable to be put into the knowledge library?
- How can we verify if the software architecture meets all requirements?

Extensive case studies have to be performed to validate the soundness of the framework. Tool support will be essential to carry out the case studies.

7 References