

Software Requirements for Architected Systems

Elena Navarro
Department of Computer Science, UCLM,
Albacete, Spain
enavarro@info-ab.uclm.es

Isidro Ramos & Jennifer Pérez
Department of Information Systems and
Computation, UPV, Valencia, Spain
{iramos | jeperez}@dsic.upv.es

Abstract

Recently, increased attention has been paid to how to establish and strengthen the relationships between requirements and architectural design. In particular, how the process must encompass changes to requirements over time and their effects upon a system's architecture. This paper sketches our work-in-progress in this field, which we concern about a methodology to guide the architecture development from the inception.

1. Introduction

From the software inception to delivery, the development process is a complex one and often involves a series of stages. A Requirements Engineering(RE) process must show how to acquire, analyze and document requirements, i.e., focusing on the customer-defined services and constraints. RE establishes the foundation on which the system-to-be should be implemented and gives support for requirements validation and evolution over time.

Architectural models have a lower abstraction level than requirements, being closer to the end system, and they must be consistent with defined requirements in order to produce a valid solution. Developing precise software architectures from an structural and dynamic point of view and been able to meet changeable requirements, is not an obvious issue [7]. A guided process appears as the best approach to achieve this goal, leading to an increased productivity and a quality solution.

2. Our proposal

In order to guide the architects, a methodology ATRIUM (Architecture generaTEd from Requirments applying a Unified Methodology) is proposed in this work(see section 2.3). It is flexible enough to deal with concurrently evolving requirements artefacts (see section 2.1) and architectural models (specified with PRISMA [8], section 2.2) in the way Nuseibeh proposes [7],

acquiring and maintaining complex relationships between them.

2.1. The emerging Artefacts in the RE phase

In the context of Requirements Engineering, goal-driven [3] and scenario-based [5] approaches have proven being useful to elicit and define requirements. The former describes the high-level-objectives that system-to-be should achieve; additionally, it provides a way of directly establishing quality attributes as goals of the target system. The latter is proposed for describing and reasoning about large-grained behaviour patterns in systems, as well as the coupling of these patterns.

ATRIUM combines these two approaches in order to overcome some of their deficiencies and limitations when used in isolation. This union help us to define not only functional requirements but also non-functional requirements, which offers us a meaningful advantage in terms of software quality [6]. This coupling is similar to the CREWS l'Escritorie approach [9] where scenarios are used to operationalise goals. It provides us maintainability and traceability among both models.

2.2. PRISMA: a model for dynamic software architecture

PRISMA has a powerful semantic expressiveness for modelling dynamic software architectures using visual languages, hiding the underlying use of formalisms and allowing the automatic generation of code. In PRISMA, types specifications include a set of elements, which are first order citizen: components, connectors, interfaces and aspects. Components are compositional units that provide the system with specific functionality. Connectors, defined in terms of interaction among components, provide a high cohesion and a loose of coupling. Their description is accomplished by a dynamic gluing of aspects (functional, distribution, coordination, etc.) following the Aspect Oriented Software Development approach, in a high abstraction level, in order to facilitate maintenance and reuse.

The types are defined using a Component Definition Language that supports reflexive properties in order to

fulfil highly-dynamic architectures. It allows to reify any type property to the meta-level in order to achieve an improved performance. Furthermore, a Configuration Language is introduced to specify the architectural model defining the topology and the instantiated types. This language has also reflexive properties allowing a non-expected dynamic reconfiguration at run-time.

2.3. ATRIUM

ATRIUM supports an iterative and incremental process that entails a set of 5 steps. They guide from an informal set of requirements, the called starting point, to an instantiated PRISMA model, where every output artefact provides an input for the others:

Step 1.- Define Goals model

In this step, the set of goals to be accomplished by the software system, will be defined, establishing both functional and non-functional requirements by using a graphical notation similar to the one used in [2]. This step is specially relevant to the quality attributes, which the system-to-be will exhibit, as they are defined as goals. This definition will be translated to 3APL [4], whose interpreter will help us to validate the definition. The model will be refined at the next iteration aided by the analysis of the Scenarios Model.

Step 2.- Define the Scenarios model

It is used to assign the different responsibilities of the elements in the system. The temporal sequences of interaction events will be represented by means of Message Sequence Charts (MSC) [10] because they are more expressive than other approaches, such as sequence diagrams. Use cases will provide a visual metaphor notation for scenarios. A scenarios-based approach will identify the possible ways to use a system in order to achieve some desired functions or accomplish implicit purposes.

They are intended to operationalise the goals defined in the previous step and, mainly, to discover new ones and other component goals because they are not as straightforward as to be captured at once. Moreover, it offers the advantage to couple both models.

Step 3.- Define collaborations

The third step is composed of the two main subtasks: identify collaborations and build and refine the message sequence chart. The collaborations will realize the use cases through message sequence charts and collaboration diagrams. The objective is to obtain a topology prototype for the concrete system by using related architectural styles and the architect ability.

Step 4.- Formalization

A set of derivation rules are provided to generate a formal specification from scenarios, goals and collaborations, in order to encourage and speed up the formalization process. This specification could be

validated and used for its automatic compilation.

Step 5.- Compilation

Using the formal model and a set of heuristics, the translation from the requirements model to an instantiated PRISMA model will be accomplished. This architectural model will be compilable into a concrete target system preserving its reflexive properties. This step will be assisted by the engineer, following an iterative refinement process. As ATRIUM is intended to be iterative and incremental, a feedback will be provided to the Step 1. In this way, it allows to have the overall models up-to-date, all over the process.

The main advantage shown by this approach with respect to others [1] is related to the guided and semi-automated process support given from requirements to architectures. Also the coupling of requirements artefacts allows a better assessment of the design alternatives. Moreover, due to the traceability among requirements and architectures, along with the reflexive properties of PRISMA, running-systems can evolve over time while remaining compliant with their requirements.

ACKNOWLEDGMENTS

This work is partially funded by the Spanish CICYT project MetaSIG (TIC2000-1106-C02-02).

3. References

- [1] J. Castro and J. Kramer (eds): *Proceedings of First International Workshop STRAW'01*, in 23rd ICSE 2001.
- [2] L. Chung, B. A. Nixon, E. Yu and J. Mylopoulos, *Non-Functional Requirements in Software Engineering*, Kluwer Academic Publishing, 2000. 472pp.
- [3] A. Dardenne, A. van Lamsweerde, and S. Fickas: "Goal-directed Requirements Acquisition". *Science of Computer Programming*, 1993, 20:3-50.
- [4] K.V. Hindriks, F.S. de Boer, W. van der Hoek and J-J.Ch. Meyer: "Agent Programming with Declarative Goals". *Procs. of the 7th International Workshop ATAL'2000*.
- [5] N.A.M. Maiden. "CREWS-SAVRE: Scenarios for Acquiring and Validating Requirements, Automated Software Engineering", vol. 5, 1998, pp. 419-446.
- [6] E. Navarro, I. Ramos and J. Pérez: "Requirements and Architecture: a marriage for Quality Assurance". 3rd IEEE International Conference on Quality Software (QSIC 2003), 2003, Beijing, Sept. 25-26. (Submitted).
- [7] B. Nuseibeh, "Weaving the Software Development Process Between Requirements and Architecture", STRAW '01, 2001.
- [8] J. Pérez, I. Ramos, J. Jaén and P. Letelier: "PRISMA: Development of Software Architectures with an Aspect Oriented, Reflexive and Dynamic Approach", Dagstuhl Seminar N° 03081, Report N° 36 "Objects, Agents and Features", H.-D. Ehrich, J.-J. Meyer, M. Ryan (Eds.), 2003.
- [9] C. Rolland, G. Grosz and R. Kla: "Experience with Goal-Scenario Coupling in Requirements Engineering", IEEE Int. Symposium on RE, June 7 - 11, Limerick, Ireland, 1999.
- [10] The next major revision of the Unified Modeling Language™, <http://www.u2-partners.org/>