Requirements, Architectures and Risks

Wojtek Kozaczynski
Rational Software, USA
wojtek@rational.com

Abstract

Modern software development processes, like the Rational Unified Process, prescribe iterative approach to software development. One of the fundamental assumptions of an iterative process is that system requirements don't have to be completely understood to commence development.

At first glance the assumption that one can start developing a system without completely understanding its requirements seems paradoxical. However, upon closer inspection requirements can be divide into many categories one of them being the "architecturally-significant requirements". It is the understanding of these requirements, the associated development risks, and the system architecture that drive the early iterations of system development.

1. Requirements Discipline In An Iterative Development Process

The Rational Unified Process (RUP) is the most widely used modern, iterative software development process. The process can be described in two dimensions: *time* and *content*, which is shown in Figure 1.

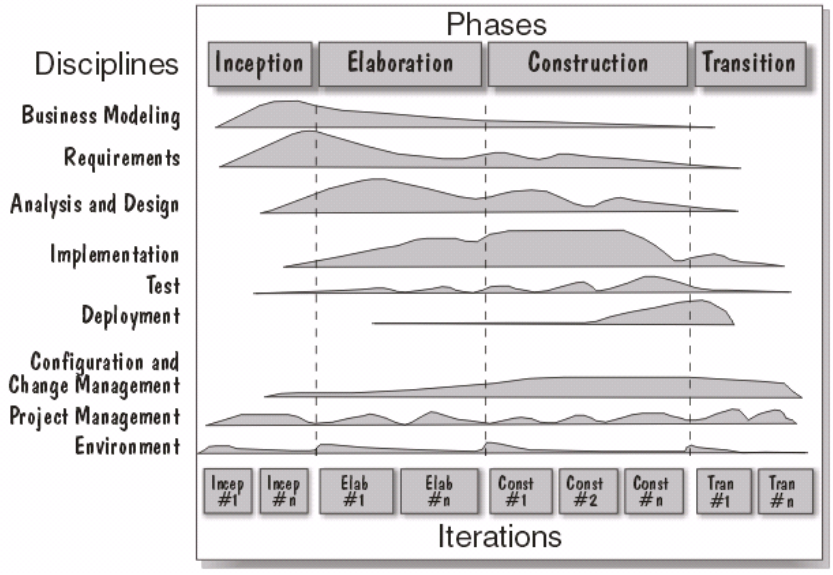

Figure 1. Two Dimensions of a Modern Software Process

The horizontal axis, described in terms of phases and iteration, represents *time* and shows the life cycle aspects of the process. The vertical axis represents *content* and shows the disciplines that logically group the process activities.

The "humps" in Figure 1 represent the relative emphasis of the disciplines over the development lifecycle. The *Requirements* discipline is a logical grouping of the activities, artifacts and roles that have to do with elicitation, description and management of the software requirements of the system under development.

As the shape of the requirements hump suggests, the definition of system requirements is not something that happens once at the beginning of the project. Actually, it would be naïve to assume that one could fully understand and document system requirements up-front. In practice key requirements are not fully understood until the architecture is baselined at the conclusion of the Elaboration phase.

This presents an interesting challenge to the architect: How is he going to come up with an architecture without fully understanding the system requirements? This is a paradox that all architects must face.

2. Architecturally Significant Requirements

RUP defines a requirement as follows:

A requirement describes a condition or capability to which a system must conform; either derived directly from user needs, or stated in a contract, standard, specification, or other formally imposed document.

There are number of ways requirements are grouped or classified. The most common grouping separates the *functional* from *non-functional* requirements, where functional requirements are those related to system use cases, while the non-functional requirements are such things as performance, reliability, supportability and usability. Another

Proceedings of the IEEE Joint International Conference on Requirements Engineering (RE'02)
1090-705X/02 $17.00 © 2002 IEEE

common grouping divides the requirements into *design*, *implementation*, *interface* and *physical* requirements (or constraints.)

If we were to write even a small sample of requirements of a real system we would notice, that they are very diverse ranging from things like internationalization, to assumptions about reuse of software components to common mechanisms to performance, and so on.

The requirements that the architect is most interested in are commonly referred to as *architecturally-significant requirements* (or simply *architectural requirements*). These are the requirements that have the most impact on the architecture, and they can come from all of the above listed requirements categories.

Even experienced architect and architecture teams often find it difficult to identify and isolate the architectural requirements, which is the main subject of this presentation.

3. Architecture, Requirements and Risks and Iterative Architecture Development

The "art" of successfully coping with the incomplete requirements paradox is based on understanding an important interplay between the iterative architecture development, requirements and risks.

Iterative development is based on the assumption that no system can be developed at once. The interactions between system components are too complex to design and analyze all of them up-front. Also, some of the component/system properties like memory consumption or performance, cannot be analyzed in any other way, but empirically. Hence, some of the system components have to be built before others can be fully designed. The very objective of the Elaboration phase of the RUP process is to come up with baseline architecture that is a mixture of an implementation (often referred to as *architecture prototype*) and a design of the critical system components and mechanisms.

An obvious consequence of the iterative development assumption is a question: In what sequence should we design-build-test the architecture? The answer is in Figure 2.

Let's start from a very useful definition of a risk:

A risk is a requirement in context.

This definition asserts, that all risks are determined by interpreting requirements in the context of the project's technology, staffing and skills and schedule. It also implies, that each risk is associated with a requirement.

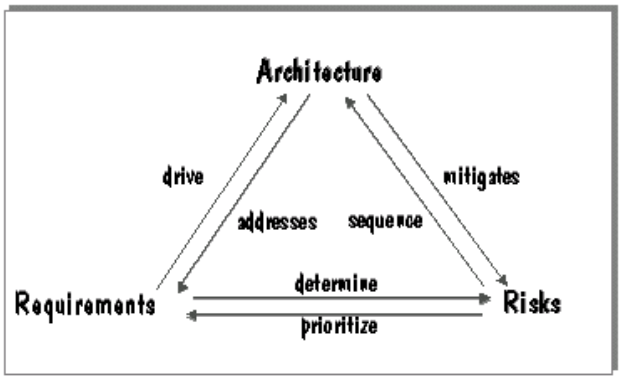

Figure 2. Relationships Between Requirements, Risks and Architecture

On the other hand, the main objective of the architecture development is to mitigate the major risks. What are the major risks? Well, we know the answer to that question; the risks associated with the architectural requirements. Hence, understanding of the risks both prioritizes the requirements and helps define the number and content of the Elaboration phase iterations.

To close the loop on Figure 2, the architecture must mitigate the major risks, as already mentioned, and address the architectural requirements.

Bottom line, the architect develops the architecture starting from the architectural requirements associated with key risks. The prioritization of the risks, and therefore the requirements, determines the sequence of architecture development.

The above implies, that the architect does not have to wait until all, or even most, of the requirements are know. He can start designing the architecture as soon as he has identified and prioritized the architectural requirements and associated risks. In addition, as he moves forward he refines and re-prioritizes the requirements and then plans subsequent iterations accordingly.

The presented argument explains how architects can cope with the incomplete requirements paradox, but it begs the question, how to discover architecturally significant requirements? This will be discussed in the presentation.