Chapter 70
High-Level Modeling to Support Software Design Choices

Gerrit Muller
Buskerud University College, Norway

ABSTRACT

The IT industry is suffering from severe budget overruns and ill-performing IT services. Some of the problems that have caused IT project disasters could have been anticipated in the early project phases and mitigated in the project follow-up by modeling the system context and the software design. This chapter shows how to make models of varied views and at varied levels of abstraction to guide software design choices. Models of the enterprise provide understanding of the objectives. Models of the specification provide understanding of system performance and behavior. Models of the design provide understanding of design choices, such as the allocation of functions, resource usage, selection of mechanisms for communication, instantiation, synchronization, security, exception handling, and many more aspects. High-level models are simple models with the primary goal to support understanding, analysis, communication, and decision making. The models have various complementary representations and formats, e.g. visual diagrams, mathematical formulas, and quantitative information and graphs. Model-driven and model-based engineering approaches focus mostly on artifacts to analyze and synthesize software and hardware. High-level models complement model driven approaches by linking the system context to more detailed design decisions. High-level modeling as discussed in this chapter is based on research performed in industrial settings; the so-called industry-as-laboratory approach.

INTRODUCTION

The success rate of large scale IT projects is disastrously low. Regularly, failures of IT projects of tens of millions of Euros are reported. McManus and Wood-Harper (McManus 2008) analyzed a set of IT projects and possible causes for failure. They note that these projects are inherently large and complex, that often stakeholders' communication and management is insufficient, that projects often fail to meet customer expectations, and they observe an overreliance on project and development methodologies.

DOI: 10.4018/978-1-4666-4301-7.ch070

Concrete examples of failed projects are large web-based health care or travel booking systems that are improperly dimensioned and hence cannot cope with the actual load. In both cases, the development focus has been on functionality and software mechanisms, ignoring operational performance needs. Estimation of operational needs and performance of the selected solution could have signaled performance problems in the early project stages.

In this chapter, we show a high-level modeling technique that, in a short amount of time, creates insight in the problem space and the consequences of options in the solution space. It is called high-level modeling because we strive for understanding and identification of issues. High-level modeling is on purpose simplifying problems and solutions at the cost of accuracy. However, even with limited accuracy a lot of understanding can be created. Critical qualities may have to be modeled later in much more detail to provide a proper confidence level.

The term "high level" refers to the amount of abstraction, and hence simplification, that is used in the high-level models. Figure 1 shows a number of high-level models mapped on a pyramid representing the number of details in the models. The pyramid figures are elaborated in Sections 2.4 and 4.1 of (Muller 2011). These high-level models are simplified so far that "manual" analysis and reasoning by humans can be done.

One example of a failed IT project is the travel management system at a large multinational company. The development of this system was outsourced to a renowned IT house. The system was tested in one of the departments before it was rolled out in the entire company. Once it was introduced throughout the company the response time degraded to hours, rather than seconds as expected. It is our experience that a few low level software design choices, such as the granularity of transactions or locking, or the granularity of instantiations and notifications, can make or break the performance of the overall design. We claim that the consequences of these choices can be predicted early in the design by a combination of simple models and measurements.

Typical for the IT domain is that architecture itself is partitioned into three layers: enterprise architecture, information architecture, and systems architecture. Enterprise architecture focuses on people, processes, and enterprise business, functionality, and performance. Information architecture describes the underlying information flow and structure. Systems architecture addresses among others partitioning, function allocation, interfaces, dimensioning, and communication protocols of hardware and software components.

Figure 1. Positioning high level models in a pyramid representing the number of details