Sidebar – Software Architectures

Patricia Lago and Hans van Vliet
Vrije Universiteit, Amsterdam, The Netherlands \{patricia, hans\}@cs.vu.nl

Software architecture is becoming one of the central topics in software engineering. In early publications, such as [Sha88], software architecture was by and large synonymous with global design. In [SG96] we read “the architecture of a software system defines that system in terms of computational components and interactions among those components”. In a traditional software engineering process, during the software design phase the system is decomposed into a number of interacting components (or modules). The top-level decomposition of a system into major components together with a characterization of how these components interact, was considered as the software architecture of the system under development. In this respect requirements engineering is an activity focussing very much on the problem space, while the subsequent design phase focuses on the solution space. We call this a pre-architecture development approach. Here, few persons (also called stakeholders) are typically involved (like for example the project manager, robotic expert, the software analyst, few developers). Iteration involves functional requirements only: once agreed upon, these are supplemented with non-functional requirements to form the requirements specification used as input for design. In particular, there is no balancing between functional and non-functional requirements.

Conversely, modern software engineering is moving to an architecture-centric development approach. Here, the discussions involve multiple stakeholders: the project manager, different classes of robotic and software experts, future maintainers of the system, owners of other inter-operating systems and services. Iteration concerns both functional and non-functional requirements. In particular, architecting involves finding a balance between these type of requirements. Only when this balance is reached, next steps can be taken.

In the latter approach, software architecture has to bridge the gap between the world of a variety of, often non-software-technical stakeholders on one hand – the problem space –, and the technical world of software developers and designers on the other hand – the solution space.

Software architecture hence describes much more than just the components and the interactions among them. It serves three main purposes [BCK03]:

© Springer-Verlag Berlin Heidelberg 2007
- It is a vehicle for communication among stakeholders. A software architecture is a global, often graphic, description that can be communicated with the customers, end users, designers, and so on. By developing scenarios of anticipated use, relevant quality aspects can be analyzed and discussed with various stakeholders.
- It captures early design decisions. In a software architecture, the global structure of the system has been decided upon, through the explicit assignment of functionality to components of the architecture. These early design decisions are important since their ramifications are felt in all subsequent phases. It is therefore paramount to assess their quality at the earliest possible moment. By evaluating the architecture, a first and global insight into important quality aspects can be obtained. The global structure decided upon at this stage also structures development: the work-breakdown structure may be based on the decomposition chosen at this stage, testing may be organized around this same decomposition, and so on.
- It is a transferable abstraction of a system. The architecture is a basis for reuse. Design decisions are often ordered, from essential to nice features. The essential decisions are captured in the architecture, while the nice features can be decided upon at a later stage. The software architecture thus provides a basis for a family of similar systems, a so-called product line. The global description captured in the architecture may also serve as a basis for training, e.g. to introduce new team members.

We do not want here to give yet another definition of what software architecture is or should be. For the interested reader [SEI06] provides a quite complete overview of what is meant by software architecture around the world.

Recently, we observed an interesting evolution of this widely discussed concept. In a more traditional connotation Software Architecture (SA) is con-