Software Architecture and Reflection

Dilip Soni
dilip@scr.siemens.com
755 College Road East
Princeton, NJ 08540
(609) 734-6545

We are exploring the emerging area of software architecture. Software architecture provides a unifying theme for the engineering and management compromises that must be made to complete a product. By relating form and function, software architecture provides the central element around which many aspects of a project can revolve. The choice, development, and refinement of a software architecture are thus critical to the software development process.

We have taken an empirical approach [1], surveying the description, analysis, and use of software architectures within Siemens. We are also investigating the different strategies for system decomposition and interconnection that are important within specific application domains.

In our survey, we have observed that the software architecture of a system can be viewed from many different perspectives [2]:

- Code architecture organizes code into files, directories, class libraries, and program libraries.
- Module interconnection architecture describes the logical structure of the system, organizing the system into subsystems, modules, and layers with details of exported and imported interfaces.
- Execution architecture describes the dynamic structure of the system with assignment of logical modules to processes and processors.
- Conceptual architecture describes the system in terms of components representing its major design elements, and connectors representing interfaces and interaction protocols between components.

These architectural perspectives capture different yet inter-dependent sets of design decisions made by different teams in different phases. For example, the code architecture is influenced by the implementation language and the development environment. The execution architecture, on the other hand, is influenced by technological support for communication, distribution, and integration.

The objective of our research project is to develop methods and tools to support:

- rigorous notations to describe architectures from a variety of perspectives,
- analysis techniques to determine and predict non-functional properties (such as performance, schedulability, configurability, and inter-operability,) and

- implementation techniques to assure such non-functional properties.

Relevance of reflection

In our opinion, there are at least three issues related to reflection:

- what system aspects (such as structure or behaviour) are to be represented and operated upon,
- how a system can reflect upon its own aspects, and
- when a system reflects upon itself (e.g., design time, compile time, installation time, start-up time, or run time).

Our work mainly addresses the first and the third issues. We have only considered scenarios where the architectural perspectives are created, operated upon, and modified as a part of the development process. Tools supporting such scenarios improve the configurability of a product in response to changes in requirements and technology.

However, their ability to support dynamic reconfiguration of a system is limited, requiring the use of application-specific architectural primitives and implementation techniques. We hypothesize that reflection facilitate development of general techniques to improve dynamic reconfigurability of software architectures.

References

[1] Dilip Soni, Robert Nord, Liang Hsu, Paul Drongowski, "Many Faces of Software Architecture," in Proc. of Workshop on Studies of Software Design, Technical Report, Dept. of Computing and Information Sciences, Queens University, April 1993.

[2] Dilip Soni, Robert Nord, and Liang Hsu, "An Empirical Approach to Software Architectures," to appear in the Proc. of the 7th International Workshop on Software Specification and Design, Dec. 1993.