ECOOP'98 Workshop W2: Techniques, Tools and Formalisms for capturing and assessing Architectural Quality in Object-Oriented Software

Track on
"Architecture in Object-Oriented Systems: Structure or Multi-layered approach?"

Summary of track discussion by group chair

The discussion initially adopted a strategy whereby some clarification of the relevant terms will be attempted, then the topic of architecture itself will be discussed so as to identify a set of issues. It was agreed that some outlining of the most pressing issues under this topic will constitute a satisfactory outcome of the discussion.

Terms

The group saw the terms Architecture, Structure, and Architectural Style as needing some clarification, or at least minimal agreement. Earlier presentations attempted to clarify those terms. In the light of these, few understandings were agreed:

1) Structure describes software modules and their interconnections. Software structures are at a lower level of abstraction than software architecture.

2) Architecture should be a conceptual representation of software at a higher level than single modules and their interconnections. Architecture could show generic categories of software elements. Software architecture should be coarser-grained.

3) The term Architectural Style was not discussed to any great extent, but it was pointed that a multi-layered architecture is one such architectural style, which can have many instances that conform to the layered view of software.

4) Software generality is different from its flexibility: Generality implies achieving certain evolvability of the software by anticipating change. Flexibility implies the ease with which new functionality can be added.

The discussion moved in the direction of the process of architecting, its critical variables and the fundamental role that software architecture needs to perform if it is to be of use beyond merely illustrating the way in which modules are interconnected to make up a software system.

The process of architecting

1) The process of architecting software, and the information system surrounding it, needs to be made explicit. One problem with current practice is that the architecting at is largely implicit. Also, such a process needs to be pragmatic in operational terms to guide software developers working real projects.

2) There is a critical role of a learning process of some kind, within the process of architecting software.

3) Architecture should be assessed statically, with a prominent role for the semantic evaluation of architecture, with respect to a particular domain, during that assessment. Assessment and evaluation of architectures needs to be explicit and distinct from other software development activities. This, as well as the role for the learning process pointed to above, stresses the importance of developing tools and techniques for performing architectural evaluation.

4) More generally, techniques for evaluating architecture should be always applied. The following techniques were pointed out:

• Scenarios
• Mapping of Domain and requirements concepts to architectural design models
• Assessment of impacts of change

5) It is vital to chart the road from a representation of requirements to the derivation of a software or a system's architecture.

The role of architecture

1) One major role for software architecture is to support the evolution of systems. It was argued that architecture gives rise to evolvability, while that does not mean that the architecture itself needs to evolve. It could be argued that stable architecture gives rise to a multitude of evolution paths for the system itself.

2) One possible direction for allowing systems to evolve is to study types of volatility (i.e. ask about key volatility areas) as well as critical variation points of the domain during analysis.

3) Methods for software development embody specific architectural styles within them, this should be borne in mind when selecting a particular software development method.

Groundwork

1) There is a need for the development of domain change typologies. Some change will be small and expected, others will be drastic and unexpected. It is useful to study domain changes in terms of categories of such changes. This will support the development of generic architectural models.

2) Techniques from Risk Management could be employed in investigating the domain/ context surrounding the project to establish key variability areas. This information should be used in establishing software architectures.

3) A distinction needs to be made on the levels of which Architectural issues need to be addressed during software development, these are:
• The Analysis level: to establish rates and categories of change.
• The Design level: to establish a top-level architecture of the software artefact that can be correlated to the analysis results (or requirements) as well as implementation level.
• The implementation (programming) level which is concerned with the technical detailed structure of the software product.

Summary of identified issues:

1) Distinction between structure, architecture and architectural style.
2) The progression from requirements to architectures.
3) Domain modelling to inform architecture.
4) Methods for evaluating and assessing architecture.
5) The architecting process.

Summary by: G H Galal (University College London)
24th August 1998