

Architecture Reconstruction using Meta-Models

Jean-Marie Favre

*ADELE Team, Laboratoire LSR-IMAG,
University of Grenoble, France
<http://www-adele.imag.fr/~jmfavre>*

One important issue in reconstructing the architecture of large scale software is that there is no agreement about what does the architecture term means exactly. In this presentation we advocate that **using Meta-Models¹ could provide significant insights in architecture recovery, both at the theoretical and practical levels.**

In our approach these meta-models are not only used to describe the architectural (meta)concepts at a very high level: they are also used (1) to ensure the link between the architectural level and the actual implementation, (2) to derive actual reverse engineering tools and (3) to define and control the process of architecture recovery.

Our experience in industrial settings shows that meta-models rapidly become complex. One way to cope with this complexity is to structure the meta-model space. We use a three dimensional space for that.

- The first dimension is related to the level of abstraction: for instance a architectural metal model is more abstract that a meta-model of source code.
- The second dimension makes the distinction between static and dynamic information.
- The third dimension is related with refinement: the highest level focuses on concepts, the lowest level focuses on actual representations².

We applied this approach in various settings and these dimensions revealed to be much more orthogonal that we expected. We also discover the need to further structure the first dimension into subdomains based on stakeholders and found useful to describe these domains as (meta) use-cases.

Our experience shows that using meta-models is also a very powerful way to compare existing approaches and tools in architecture reconstruction, to implement new tools and to define methods in a precise way. In particular the scenarii provided by L. O'Brien, C. Stoermer and C. Verhoef in "*Software Architecture Reconstruction: Practice Needs and Current Approaches*" (SEI) could be further refined thanks to meta-models while keeping a great level of generality.

-
1. As an illustration a typical meta-model for software architecture might include classes like *System* and *Modules*, and associations like *DependsOn*. Simply put a meta model is sometimes correspond to the notion of schema, of domain in Rigi, of DTD in XML, etc. One important difference is that thanks to UML a lot of information can be associated to meta-models, including for instance semantic constraints described in OCL.
 2. This dimension is somewhat similar to the difference between the PIM and PDM in Model Driven Architecture (MDA from OMG). One important difference of our approach is that we focus on a reverse engineering, not on only forward engineering.

We strongly believe that meta-model engineering and meta-model reverse engineering is an activity that can really fit the needs of software reconstruction in industrial context. In particular a very good property of meta-models, is that while the level of abstraction raises, the technology necessary to support the approach does not necessarily change. In other words most of the tools and techniques currently used can be revised or even directly applied at the meta-model level¹! We develop both specific and generic exploration tools to construct architectural views on software suited for a particular set of stakeholders. An environment like G^{SEE} , the Generic Software Exploration Environment, is driven by a meta-model. It enables “on-the-fly” reverse engineering of meta-models. It also support the exploration of both architectural models and architectural meta models, all this at the same time and without any kind of change in the tool. These practical results have been made possible only by recognizing the existence of meta-models (they are often implicit) and by using a suitable meta-meta model to describe and implement meta-models.

Our approach is the result of the experience gained during a seven-years case study. This case study was led in the context of a collaboration between the University of Grenoble and Dassault Systèmes, the CAD/CAM world leader and one of the largest software company in Europe. Recovery meta-models was a difficult yet very interesting task since Dassault Systèmes developed a lot of proprietary yet very powerfull technologies to deal with architectural issues related with its software (more than 1000 software engineers evolve at the same time a product line made of more than 50000 C++ classes and 8000 components based on a proprietary component model).

We developed different tools and methods in this context, but the most promising result from our point of view is that the same approach could be applied in other contexts. **Working at the meta-model level is not only a challenging task from an intelectual point of view; if the link with actual problems and implementations is maintained this approach provides a very high degree a generalization. This is very useful to handle very specific situations such as the one found in Dassault Systèmes.**

1. In fact this result should not be surprizing since languages like UML show that a single language (and associated tools) can be used to described instances, model, meta-models, meta-meta models, etc/