Session 3: Product Line Development Experience II

Dewayne E. Perry
Software Production Research
Bell Laboratories
Murray Hill, NJ 07974

Wilhelm S. Schaefer
Computer Science Dept
University of Paderborn
D-33095 Paderborn, Germany

Lawrence G. Votta
Software Production Research
Bell Laboratories
Naperville IL 60566

Schaefer introduced Stephen Doublait of SODALIA as the keynote for the second session on experience with product line development. Doublait's focus was on reuse within SODALIA. He began with the question: why bother? On the one hand we have ad hoc, opportunistic code reuse; on the other hand we have systematic, planned artifact reuse. The former is historically the typical approach taken by smart, open minded programmers. The latter requires investment in enabling processes and technology, and offers the advantage of being less skill-sensitive. What is not clear is the extent to which the reliance on less skilled developers is scalable.

Some of the proven reuse enablers include top management commitment, business motivation, process oriented software development, organizational cross-project support, and use of object-oriented techniques throughout the life cycle. At SODALIA, management commitment is illustrated by the fact that there are 12 architects out of a group of 200 dedicated to reuse. Balzer claimed that reuse comes from mapping the domain onto modules. Doublait indicated that the OO techniques were the least important factor, but that they were a very efficient means for conveying the domain model.

Of critical importance is a corporate-wide reuse strategy. How do you procur reusable assets? How do you make these assets available? How do you use what is available – that is, how do you support software development with reuse. For the software reuse asset library, SODALIA uses a classify and insert, retrieve and reuse model. Osterweil noted that if you had artifacts produced by a different process you would have artifacts with different characteristics and could very well not match up to the reuse intentions. Doublait responded that the reusable assets were such things as requirements, designs, code and tests with an emphasis on function and quality.

Doublait provided a taxonomy of reusable components. Perry noted that it looked mostly like functional components. Doublait responded that they wanted to include functional, technical, quality, and operational requirements and to try to foresee all the quality requirements. Tully observed that it is much easier to find functional requirements than quality. Perry noted that SODALIA's experience sounded like they had a relatively uniform set of quality requirements. Doublait conceded that this was true, although there was some range of quality requirements, and that it was easier to change functionality.

5 case studies were done on libraries containing small components up to medium size components. There are on the order of 10-20 internally and externally developed libraries and there are virtual components that can be bought. Balzer correctly pointed out that at that level you don't need a tool for component retrieval and use.

There are three aspects of measuring progress: how to make components reusable (reusability metrics), how to make components available and how to benefit from software reuse (reuse economics metrics).

The discussion then turned to how you get benefit from reuse. It works well where you have components that mix and match. Conradi noted that he and his group have struggled with productivity measures and that they work easier when the framework is the smaller. Balzer rightly pointed out that the real payoff comes from reusing larger components. In response to Madhavji's question as to how one institutionalizes the measurement program, Doublait indicated that nothing formal had been done but that the program works fine.

Bandinelli asked about generic and customization versus straight reuse. Doublait responded that all the components were directly reused. It is basically a framework with high adaptation. Unmaintained components are used only reluctantly. Tully noted that one usually had traceability problems with the various levels of reuse (for example, requirements and code). To what extent does that affect genuine or safe reuse

and are the traceability documents covered?

Doublait defined asset reuse maturity levels as opportunistic, adaptive, planned, domain-specifically architected, and generative. In the reuse reference model, within each domain one finds product families. Vertical reuse is that from one version to the next and horizontal reuse is between domains. He also mentioned the notion of diagonal reuse which was the reuse of product in domains.

Perry noted that there seemed to be confusion as to what was use and what was reuse. For there to be a product line, Balzer claimed that there needed to be more than just random products in the same domain. There needs to be substantial sharing of assets in that domain across those products.

In domain centered reuse, the solution space is defined by enumeration with multiple applications within a single domain. Perry thought it was more complex than that in that there were likely to be a large number of domains in a particular product, especially if it is a large complex product like an electronic switch. Part of this depended on how one defined the granularity of the domains.

The rationale for the reuse program is in the business value, not in any customer value. Alex wolf asked about the problems of duplication and reuse reengineering. Doublait responded that there was little legacy code to worry about and that most of the reuse engineering was done from scratch.

In considering the problems of component ownership and maintenance, the attempt is to put bounds on who owns what and maintains it, with single ownership and maintenance preferred over multiple ownership and maintenance. Configuration mangement is provided by ClearCase. Schaefer noted that it is often useful to have an independent entity responsible for the ownership and maintenance.

Evidently there is little actual cost data — there have been very poor measurements, but the anecdotal evidence is good. There is no reward system in place – that is, there are no incentives; reuse is enforced and all artifacts are produced according to the reuse process.

The five case studies were presented and discussed. In one, the components were very expensive and used without adaption. In another the reusability criteria were established from the start and hence were known from the beginning rather than derived. In another components were designed to be used across multiple applications. While Tully claimed that this seemed like just good design, Doublait disagreed in that it was not just for one application but across multiple ones.

Finally, there is a quality assessment process by which reusers are informed where the component is on the quality scale. This process is enforced by qualification audits in which it has to be shown that the process has in fact been followed.