Parametric and provisioning approaches are again obviously useful for various kinds of generic descriptions and provide the most direct means of deriving their product architecture from the product line architectures.

8 Summary

I have considered a variety of useful ways of 'genericizing' architectural descriptions (or prescriptions). I claim that a generic architecture is a fundamental requirement for a product line and that each of these approaches is needed as a means of defining some important elements in such a generic architecture.

This article was processed using the LaTeX macro package with LLNCS style

6 A Service Oriented Architecture as a Generic Architecture

One of the typical kinds of problems found in developing such large and complex systems such as telephone switches is the need to provision the various products with different features. Provisioning these systems is not the kind of thing that can be done with parametric or variation independent approaches. One can always of course do it with either styles or under–constrained descriptions, but that does not help much if one wants these provisioned features to be architectural features.

Thus an approach to describing a product line architecture is one in which the various architectural services that may be provisioned are defined as part of the architecture and are then selected in an instantiation process to define a particular product. One advantage of this approach is that the possibilities are explicit in a more tangible way than in a parametric approach. Moreover, if done properly, the architectural dependencies of these services are also made explicit and the implications of choices are thus more explicit.

As with the parametric approach, instantiation is accomplished with well–understood technology. Analysis and planning both can be done relative to the product line description with the added advantage that the planning of a specific product can be derived from the product line planning itself via the selection mechanism of provisioning.

As long as the evolution of the product line architecture is done via the addition of new services, existing product architectures will remain valid instances of the new product line architecture.

While this goes a long way towards a useful approach for provisioned products, it is likely to be insufficient in itself for a complete product line specification.

7 Putting The Pieces Together

I think it is clear at this point that a comprehensive approach to defining a generic architecture for a product line requires all of these different ways of addressing various product line issues.

Styles are certainly needed for aspects of the product line that are orthogonal to the specific component structure. For example, one may want to define a style for initialization or fault handling that must be satisfied by all the components in a product line to ensure appropriate cross–product use.

Under–constrained descriptions always provide a wider degree of flexibility than over–constrained ones. Clearly some aspects of a product line will be best served by this approach where large degrees of design and implementation freedom are useful to respond to such things as changes in technology.

The variation–independent architecture is certainly needed where you want to delay such considerations as platform or distribution until build time or even execution time.

involved. Evolution of the product line architecture implies evolution of the product architectures.

Because of the identity of the product and product lines architectures, issues of analysis and planning at the product line level apply to the product level.

The downside of this approach is that it may not be possible to isolate all the variations in this way. Certain properties such as distribution, fault–tolerance, etc may be amenable to this, but differing functionality may not be.

Another negative aspect is the standard specification problem of talking about what is not there.

5 A Parametric Architecture as a Generic Architecture

A standard approach for generalizing is that of parametric abstraction. The parameterized component is then applicable across a wide range of arguments (in programming languages defined typically by types). The limits of applicability depend on the constraints that are checked on those arguments. That partly depends on the type system and what is allowed as a first class parameter types. For example, in Ada generics, the range of types usable as parameters is larger than for functions and procedures. In macro languages there are typically no constraints at all. But then there is no guaranteed substitution safety either.

The utility of this approach is the same as for packages and operations: the architecture specification defines a family of possible instantiations and for which the properties of the product line can be ensured for the various instantiations. The variations required for each possible product in the line are well–defined and known. Moreover, the instantiation of a specific product architecture is a well–understood technology and the instance can be derived automatically from the argumented product line description.

Here again, analysis and planning are doable at the product line rather than the product level.

Evolution of the parameters may seriously affect individual product architectures. If the evolution is limited to broading the types of the parameters, or perhaps upward–compatibly extending the parameters, then the individual product architectures should remain valid.

There are two limiting factors. First the kinds of the parameters allowed may seriously affect how well the generic architecture serves to cover the necessary products. If the kinds of first class objects are too limited, then one may not have sufficient descriptive power to satisfactorily describe the product line. Second is the question of whether parameterization covers all the kinds of variation that one might need to have among the products in a product line. We have seen examples above that suggest that parametric approaches are not sufficient in and of themselves.

but one which has marrower bounds) and one will have to analyze the product architecture to ensure its conformance to the product line constraints.

This approach seems to be an appropriate one to use if the primary difference among the products is something like performance and in which the functionality is primarily the same. On the negative side, extending the product line is a significantly more constraining task. Unless you evolve the product line architecture, the new products must be definable within the current constraints. In evolving the under–constrained product line architecture, care must be taken in its expansion not to inadvertently nullify current products as constituents of the line through the addition of further components or constraints. Constraint relaxation, of course, does not cause such a problem.

4 A Variance–Free Architecture as a Generic Architecture

Again the differences between this and the preceding ones are subtle. Here the architecture is not under–constrained. It is instead a fully described architecture but one in which the variances among the products are not considered to architecturally important – that is, the product differences are an issue of design and implementation, not an issue of architecture.

This approach is useful when your product line spans a significant range of options with respect to a particular aspect. One such example is that of whether the system is centralized or distributed. If the products range from simple centralized systems through to complex multi–processor and distributed processor systems, then this characteristic of the system might well be one that you want to bury in the infrastructure and not have as an important architectural issue. In this case, you might want to have a distribution independent architecture. Distribution then becomes an implementation or even a administrative issue, but not an architectural issue.

What is interesting in this case is that there is a significant implication for the implementation to support this kind of variance independence. To make the architecture independent of issues of distribution implies a class of architectural components which will support that independence.

Another example might be platform independence. Here again, there is an implication about what the structure of part of the architecture must be in order to bury the actual platform specific aspects in the design and implementation rather than have them visible at the architectural level.

There is a significant appeal in this approach. Analysis and planning can be done at the product line architecture level. If the right product characteristics are made independent of the architecture, then new products can be derived from the product line architecture with relative ease merely by providing the appropriate implementation specific components in the design and coding phase in such a way that they conform to the product line architecture. The individual product architecture is the product line architecture; there is no derivation

specification. One primary advantage is that new products can be added to the line with ease as long as they conform to the basic product line stylistic constraints. This provides a wide degree of latitude in the the various products and what they provide relative to the core essence of the product line.

One of the negative side effects of this approach is the amount of work needed to refine the product line style into a particular product architecture. With the intent of a style as capturing only the essential architectural aspects of the product line, those aspects must be extended and added to in order to create in individual product architecture. As such the product architecture must be analyzed for conformity to the product line architecture.

As a result of this lack of completeness other aspects of architectural based development suffer as well. For example, analysis of the product line architecture will, of necessity, be less comprehensive. Project planning will be similarly less comprehensive at the product line level and the majority of planning work will be delayed until after a complete product architecture has been extended from the core style.

Further care must be taken in evolving the product line's architectural style so as not to invalidate existing product architectures. With each change to the product line style, the individually derived product architectures will have to be re–analyzed to ensure that the product architectures remain conforming to the style.

On the whole there are better uses of styles for product line architectures than defining the generic product line architecture itself. For example, one could define a set of styles defining such things as initialization, fault recovery, etc that all the various components in the architecture must adhere to.

3 An Under–Constrained Architecture as a Generic Architecture

The difference between an architectural style and an under–constrained architecture is a subtle one. The difference is fundamentally the difference in the completeness of the architectural description. A style is meant to focus on certainly critical features and isolate them from non–essential and non–stylistic features. There is no requirement for completeness of an architectural description in any way.

With an under–constrained architecture the idea is to capture the product line as completely as possible but in such a way that the variations are not ruled out by overly constraining the architecture. The variance is within the confines of the architectural constraints, not within the aspects that have not been defined.

This approach goes a long way towards solving the weaknesses of the stylistic approach in terms of analysis and planning at the product line level. Further it is much easier to create a product architecture from the product line architecture. However, it is still not a simple matter to produce the product architecture from the product line architecture (it is still primarily a creative process as with styles

Generic Architecture Descriptions for Product Lines

Dewayne E. Perry

Bell Laboratories, 600 Mountain Avenue, Murray Hill, NJ 07974 USA
dep@research.bell-labs.com, www.bell-labs/com/usr/dep/

1 Introduction

Two of the fundamental needs in defining an architecture for a product line are

- to be able to generalize or abstract from the individual products to capture the important aspects of the product line and
- to be able to instantiate an individual product architecture from the product line architecture.

In other words, having a product line implies having a generic architecture from which the individual product architectures can be derived in some manner.

There are a number of different ways in which one might go about defining the product line architecture so that this desired level of genericity can be achieved. Five possible ways of doing this are

- use a software architecture style,
- use an under–constrained architecture description,
- define a variance–free architecture,
- use parametric descriptions with varying binding times, and
- use a service oriented description for selective provisioning.

In the end, I think you will need all of these for a systematic and complete generic product line architecture. I will discuss each of these in turn and delineate their strengths and weaknesses.

2 A Style as a Generic Architecture

There is a certain intuitive appeal in using a product line specific architectural style as the generic architecture for a product line. It would capture the essential characteristics of the product line while ignoring the variations and leave them to be supplied as needed in the actual product architecture. These essential characteristics would encompass the necessary components that each instance must have, the basic minimum interactions that each instance must have and the basic constraints on these components and interactions.

The utility of a style description is that it represents the minimalist approach to software architecture in general and product line architecture in specific. Only the critical aspects of the product line need to be considered in the architectural