Reference Architectures in a product line process context

Sergio Bandinelli
European Software Institute
Parque Tecnológico de Zamudio #204
E-48170 Bizkaia, Spain.
Tel: +34-4-420 95 19
Fax: + 34-4-420 94 20
e-mail: Sergio.Bandinelli@esi.es

1. INTRODUCTION

The architecture level of description of a software system provides an overall view of the system, making explicit its design rationale and constraints. Recently much effort has been devoted to software architectures, setting the grounds for a new discipline [1,2,3]. In particular, specific attention has been given to architecture description languages (ADLs) to support architectural modelling

A reference architecture is a software architecture for a produ of application systems in a given business domain. The reference architecture contains design rationale, engineering decisions, assumptions and constraints that apply to all members of the product family. Reference architectures generally consist of partially specified systems composed of generic or abstract components that are replaced by real components when the architecture is instantiated to form an actual application.

In other words, a reference architecture provides the means for structuring knowledge on how to design and implement applications in a given domain in an organisation. Once available, this knowledge can be used in building new applications. This production scheme is more or less explic in many organisations. In doing so, software managers and practitioners are facing several difficulties. These include:

- What are the benefits of having a reference architecture? How are these benefits measured?
- Is a given reference architecture appropriate for a new development? If not, what are the necessary changes?
- How do we guarantee the adequateness of a reference architecture to fulfil user requirements? How do we determine whether a given functionality is realised by an architecture?
- How is an application architecture derived from a reference architecture, without corrupting its original design principles?

At ESI we are addressing these questions in the context of our technology plan. We firmly believe that these problems need to be faced from a proc Thus, our approach is to position reference architectures in a reuse-driven product-line process to analyse possible solutions from that viewpoint. In the rest of this paper we first provide a product line process context for reference architectures. We then explore technological issues for reference architectures in the light of this process context.

2. PRODUCT LINE PROCESS CONTEXT

Traditional life cycle models have originally been conceived under the assumption that they apply to the development of an individual software. These models are generally structured in a series of phases that go from feasibility studies to product delivery and maintenance. Product line development is a different approach: instead of having separate development processes for each product, the set of products that address the same business domain are regarded and developed as members of a standardised product family.

Product line development introduces new requirements to software processes and support technology. The extent and nature of the impact that the introduction of product line development may have in an organisation's processes structure is not yet fully understood and the subject of debate and research. However, it is broadly agreed that product line development is structured into two separate processes with different scopes. Domain Engineering (DE) focuses on the development of a product family adaptable to deferred requirements and engineering decisions. Application Engineering (AE) is aimed at deriving a single product from the family to meet specific user requirements. DE is an iterative process that incrementally grows the domain. Each enactment of the AE process corresponds to the production of a new application within the

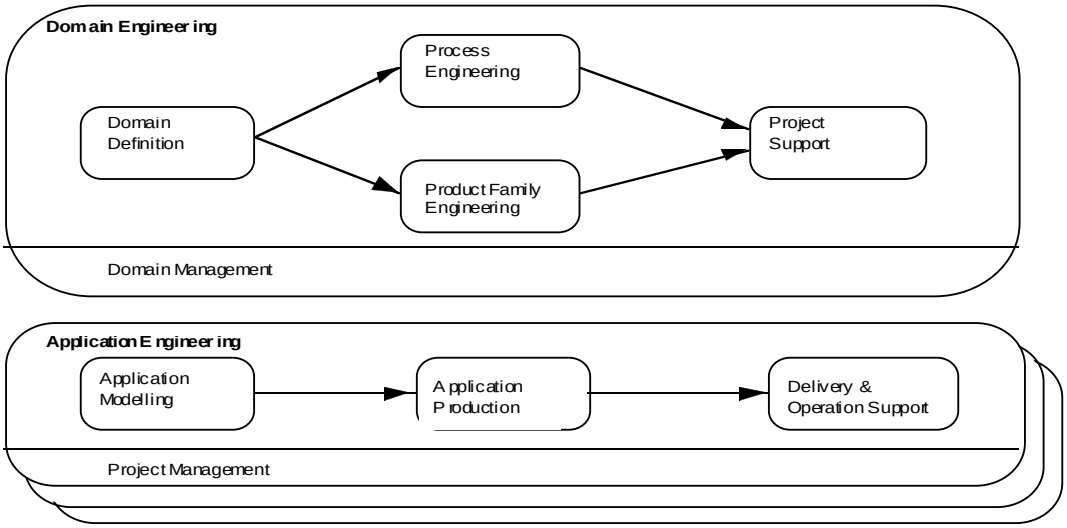

Fig. 1: DE and AE processes as specified in RSP.

Figure 1 shows the product line development phases as presented in RSP (Reuse-driven Software Processes) by SPC (Software Productivity Consortium [4]). Within the DE process, the Domain Definition establishes the boundaries of the domain and characterises all the potential members of the product line. The products in the product line address the same business domain, thus they share several common characteristics. However, the products of the product line differ among each other, since they are addressed to different users. The decision model identifies these variabilities/assumptions in the product line by defining a set of (user) decisions that are sufficient to characterise a particular product in the product line.

The product family engineering activity is similar to traditional development but, instead of developing a single product, it aims at developing the whole family of products. It goes through requirements analysis, design and implementation taking advantage of the product commonalities and leaving open the decision identified in the decision model. The process engineering activity is to establish a standard process for deriving a product from the product family. Finally the project support activity is in charge of providing support for the enactment of application engineering activities. The AE project defines decisions to match user needs by following the process developed within the process eng

3. TECHNOLOGICAL ISSUES OF REFERENCE ARCHITECTURES

The reference architecture is developed as part of the product family engineering activity within DE. Therefore, it contains open decisions regarding unresolved specific user needs. The reference architecture of a product family development, since it links user requirements with implementation components, provides the skeleton in which all other components are integrated. Each application architecture is obtained by resolving the open decisions corresponding to specific user needs. Key issues in the representation and management of reference architectures:

1. A reference architecture must be able to capture the the va product family.

2. Explicit links must relate the variability in the reference architecture to the same kind of variability in other work-products (such as requirements documents, code, test cases, etc.).

The mechanisms offered by currently used architecture description languages to capture variability are mainly those of high-level programming languages. This includes a number of language concepts ranging from the traditional function parameters, to modern parametric packages, abstract classes, inheritance and late binding mechanisms. Another approach to represent variability in reference architecture is by using a meta-language of macro directives. When resolved, the macros can be processed to obtain an application architecture representation. An advantage of the latter approach over the previous ones is that the macro meta-language can be independent of the target ADL.

Variability adds complexity to the reference architecture description. An ADL must thus offer appropriate language mechanisms to represent this variability. There are some desirable features:

- Variability is localised. The selection among different architectural variants should be localised in a single point of the representation. Otherwise, the selection of one variant may require changes in multiple points of the architectural description, which is an error-prone process.

- Variability should not obscure the architecture. Variability adds complexity to architecture representations. Therefore it is convenient that the architecture structure remains evident even if several variants co-exist in a single representation.

- Variability should be adequately captured in graphical representations. Almost every architecture description is accompanied by one or more graphical representations. Thus, the variability must also be represe

Capturing variability in the architecture description becomes of great value when this variability is linked with the other work-products in pr In a product line development process, the decision model says how the products that belong to the product family differ from each other. In a new application development, the first step is to produce an application model (see Figure 1) in which the open decisions of the decision model are resolved b the user requirements. If explicit links are maintained between architecture and these links can be used to determine which variable parts of the

reference architectures are to be used to instantiate it into the application architecture, maintaining its internal consistency and original In addition links to code components can be used to guide (or automate) the derivation of the application implementation.

Hypertext technology is a good starting point for supporting the linking of this work-product network. The explicit network links also guarantee forward and backward tracing of requirements for testing and documentation purposes. Of course, this network has to be produced as part of the domain e

The architecture level of description represents a point in which technological solutions and process issues meet a We observe that this provides an adequate context for exploring new technological solutions or combining existing ones to address the questions stated in the introduction.

4. REFERENCES

[1] David Garlan Ed., Proceedings of the First International Work
Architectures for Software Systems, Seattle (USA), April

[2] Dewayne E. Perry, Alexander Wolf, "Foundations for the Study of Software
Architecture", SIGSOFT Software Engineering Notes, vol. 17, n. 4, ACM, October
1992,

[3] Mary Shaw, David Garlan, "Software Architecture, Perspect
Discipline", Prentice Hall, 1996.

[4] Software Productivity Consortium, "Reuse-driven Software Process
Guidebook" SPC-92019-CMC, Version 02.00.03, November 199