The building block method. Component-based architectural design for large software-intensive product families
Müller, J.K.

Citation for published version (APA):
Object design starts with domain objects from application domain modelling and refines them to implementation objects (see chapter 4). Aspect design complements object design by identifying functionality which crosscuts objects. The uniform design of aspect functionality promotes conceptual integrity. Aspects are a second partitioning of a systems functionality (see chapter 5). Concurrency design describes the mapping of objects and aspects to computing resources (see chapter 6).

Composability design deals with the identification of BBs and their relations. BBs are construction elements having provides and requires interfaces. BBs are grouped in layers and have a partially ordered dependency relation. Generic BBs, i.e. component frameworks, are means for encapsulation and extension of common functionality. The architectural skeleton is formed by the set of layered generic BBs. These concepts are used in the design of product family architectures. Deployability design describes the mapping of BBs to different deployment scenarios (see chapter 7).

Products evolve through new and updated features. During commercial design a product family is defined as consisting of sets of overlapping features. These features are an important input of the BBM for creating a product family architecture. Relations between commercial features are taken as guidance for the design of inter-BB relations. Feature orientation means that the software architecture is connected to these evolving features. The quality of a component-based product family architecture will depend on its resemblance to the feature structure (see chapter 8).

The design of large software-intensive systems necessitates an emphasis on the actual construction elements. The use of the BBM leads to those construction elements, namely the BBs. A set of complementary global design models describes the position and role of the construction elements. They are the acyclic dependency graph of BBs, the various aspect designs, the concurrency design and the deployability design. They are light-weight and updating their documentation is eased by their relative independence.

Note that it is not possible to make statements about the ease or difficulty of updating an actual design. Likely evolution and extension are taken into account by using input from domain modelling, commercial design and technology choices.

The development process consists of an initial stage and a steady stage (see chapter 11): In the initial stage a single product (or a small set of products) are developed, which have a potential to serve as a base for a product family. In the steady stage more members of the family are developed. The architecture developed in the initial stage is taken as a base for the product family architecture. The
product family architecture is the basis for a long-lived product family. Refactoring is important to keep this base conceptually clean. The criterion for finishing the architectural design is that all BBs can be developed in a classical waterfall.

The BBM is based on a rich underlying architectural meta-model consisting of a domain object model, a product feature dependency model, the Building Block design dimensions, that is the object model, aspects and the concurrency model, the Building Block dependency model, and the deployment model. The first two are taken as inputs from its enclosing rational architecting model. The architectural meta-model provides the basis for expressing the relevant architectural elements of large software-intensive product families.

The BBM is presented as a core method describing the concepts of the method. A specialisation of the BBM for centrally-controlled distributed embedded systems describes additional guidelines (see chapter 10).

The BBM is based on experiences gained in the development of a product family of telecommunication switching systems (tss). Various examples throughout the text describe designs for parts of that system. The appendix contains a description of the architecture of tss and some experience data. The original intention in the development of tss was to develop a configurable product family that would achieve conceptual integrity. Reuse was achieved as a by-product [RF96] [FJ95]. The method has also been used in designing parts of a family of medical imaging systems [Wij00].

The goal, most important for an architecture, is to support the management of development complexity. Component-based approaches provide a way of extending a system but also of reducing the system, notably by removing superfluous components. Keeping an architecture up-to-date and effective is an essential precondition for high-quality product lifecycles.

The Future

The BBM can be evolved and extended in several ways.

The embedding of the BBM in the rational architecting process can be extended to a general multi-view architecting method covering the complete range of the architecting model. Such a method would address multiple viewpoints in each of the tasks, providing logical threads between these viewpoints and give patterns for selecting design mechanisms for such threads.

The relation to Microsoft’s .Net [Pla01] can be elaborated. This might be fruitful because .Net does not use a global registry like COM but uses direct
dependencies between assemblies, the name in .Net for components. A so-called manifest describes the interface of an assembly. Consequently, the BBM can be used as a design method for .Net applications.

A further point is the application of the BBM to a wider range of applications. Besides the feedback for the method it would provide more examples of specialisations of the BBM and of aspects in particular. Consolidated infrastructures for programming environments such as the intentional programming system [CE00] (mentioned in section 5.3) and server components such as COM+ [Pla99] and EJB [Mon00] provide support comparable to system infrastructure generics of which each application component can make use. The BBM can be specialised in ways similar to the specialisation for centrally-controlled distributed embedded systems (see chapter 10) to provide more support for these kind of systems.