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

Citation for published version (APA):
The initial stage comprises the product family engineering process and the product engineering process. The product family engineering process executes the process of architecting making use of the BBM. The product engineering process focuses on the specification and implementation of the specific products using the architectural design resulting from the product family engineering process to the extent that it is available.

The steady stage comprises the product family engineering process, the product engineering process and additionally the platform engineering process. The platform engineering process is responsible for the development of the architectural skeleton, that are the common BBs shared by most of the products. In the product family engineering process of the steady stage the architects determine if new features are within the tolerance of the architecture. If this is the case no architectural changes are necessary. Product engineering uses the BBs of the architectural skeleton and may develop new BBs as necessary.

The three development processes product family engineering, platform engineering and product engineering are similar to application family engineering, component engineering and application engineering of [JGJ97].

### 11.3 Building Blocks are Stable for Subsequent Phases

BBs, including generic BBs, specific BBs and system infrastructure generics, are identified in the architectural design process. The architectural design assigns functionality to BBs and defines design guidelines and constraints for BBs. A BB is developed by being specified, designed, implemented, integrated and tested (see section 11.7). A BB remains a stable entity throughout this process. Even in a deployed system the BB is recognisable. Its identity, as designed in the architectural process, will remain stable throughout the system’s lifetime. This allows tracing of BBs from component identification to component deployment.

### 11.4 Building Blocks and the Waterfall Model

BBs can be developed in parallel. Areas of risks must then be identified and solutions proposed by the architectural design process. Therefore, the process for the development of the BBs may be simple. The simplest process is according to
the waterfall model, without iterations: specification, design and coding. This should be sufficient for the development of BBs. We moreover use it as the quality criterion for the architectural process, i.e. the simple waterfall model should be sufficient to develop BBs.

*Heuristic 102*: Define and detail the architecture in such a way that BBs can be developed according to a simple waterfall model.

Technical know-how and experience are essential for achieving such an architecture.

### 11.5 Documentation

The architectural design is documented in the architecture document. Like any other documentation, it should present the logic for the architectural design rather than the historical process [PC86]. The architecture document contains only an architectural view of the system.

*Heuristic 103*: The architecture document describes the architectural models such as the BB dependency model, aspect designs, concurrency design and deployability design. The architecture document should be structured in a way which minimises the impact of changes.

This means that specification and design studies of certain areas have to be recorded in investigation documents, which are later to be replaced by the respective BB documentation. General rules and guidelines are part of the architecture document.

*Heuristic 104*: The BB documentation consists of at least three documents: its specification document, its design document and its code document.

Other documents which support other stakeholders may be documents for test cases, manuals or user interface descriptions. The BB documentation may be based on parts of the investigation documents.

Each of the documents mentioned above may be organised as a set of sub-documents which are independently handled by a document-management sys-
tem. This is especially important if parts of a document differ in evolution characteristics.

This leads to document dependencies as shown in figure 71. Feature specifications are the input for the architectural models. The architecture document and investigation studies are input for the BB specification and design. However, the complete specifications cannot be derived from the architecture document. Additionally, the BB specification has to be based on requirements derived from feature specifications.

11.6 Layered Development Processes

The architecting process and the BB development process are separate processes. They are related in that a BB can be developed as soon as it has been sufficiently defined. This leads to the notion of a layered process. One process, the architectural process, is on a lower layer and the other processes, the BB development processes, are on a higher layer (figure 72). The aim is to develop BBs in a simple waterfall model without iteration (see section 11.4).
Heuristic 105: Consider a deviation of the BB development process from the simple waterfall a quality problem of the architectural process.

A new risk, identified during BB development, is therefore signalled to the architectural process. The solution is worked out under the control of the architectural process.

The overall design process does not follow the model of phased transformations. Instead, the architectural design remains stable throughout component development.

In comparison with the spiral model [Boe87], the risk-driven initial cycles are mapped to the architectural process, while the final waterfall is taken as the BB development process. Work distribution, work parallelisation and work planning are all directed at BBs.

11.7 Incremental Integration and Test

The incremental integratability of the BBs is used for the integration and test process. Any developer will be able to integrate and test his BB on the platform of the lower BBs. The system integration is completed when the highest layer has been integrated.

Integrating and testing a BB tests the lower BBs. Achieving a stable set of BBs is easier in such an incremental manner than when all BBs are integrated at once.
Heuristic 106: Proceed with the process of integration by extending a stable set of BBs with one or a few BBs.

Starting with the BBs of the lower layers BBs are added step by step to the system to be integrated. The experience is that the time needed for the integration is shorter due to the incremental process.

Furthermore, deadlines can be set in steps according to the incremental structure, from lower layers to higher layers. Layered subsystems will usually be reasonable groupings to have the same deadlines.

Functional tests will be started during incremental integration. Tests with the complete system concentrate on long-term stability and stress conditions. Automatic regression tests and stress test are crucial for the success of evolving systems.

Incremental integratability is the main contribution of the BBM to the testing process.

11.8 Tool Support

Standardisation is the basis for automation and tool support. The BBM defines several concepts which lend themselves for this. Examples are:

- partial ordering of BB import relations,
- grouping of BB in layers,
- generics which support certain aspects, and
- component model attributes of BBs.

More standardisation is possible in the area of design, coding and development environment.

We give several examples of tools taken from tss. The tools are based on those standardised concepts.

Example: tss Architecture Support Tools

Feature descriptions, relations to other features, organisational responsibilities and various status attributes are kept in a feature database and administered via product
strategy and product progress meetings. Anybody in the product management and development departments may read the feature descriptions and write comments on them.

SMI Commands
Tables
Reports
SMI Formats
SMI Strings
SMI Types
Dynamic Sets
Processes
Memory Pools
Program Exceptions
Premature Data

Data Definition
Data Base

Code Generators
SIG interpreted
BB Code and Data

Documentation
Generators

MMI

On-line Guidance
Table Headers

Office Data

Manuals

Figure 73: DDD and Generators

The dependency relation between BBs as defined by the partial ordering is managed via a specific administration tool. The allowed dependencies to other BBs are described per BB. Dependencies may not be changed without authorisation from the architectural team.

Different forms of documentation are supported through document generators of the Data Definition Database (DDD). Figure 73 gives an overview of the DDD. Besides the code generators for supporting the system management interface (SMI) code, there are also generators for office data and for manuals. All this documentation is consistent with the system implementation by virtue of generation from a single source. The DDD tool thus supports data consistency between different departments.
Data generation for specific system configurations is supported by further database-based generators.

11.9 Organisational Consequences

Since architecture is not only a phase but a process itself, an architectural team [Kru99a] is responsible for the creation and evolution of the architecture. BB development teams may be organised according to layered subsystems [DKO*97].

Architects are not specialists in all areas [Mil85]. Figure 74 (adapted from [RM97]) shows an example of the depth of understanding required on the part of the architects. This can be achieved either by an architectural team or by architects together with chief designers who have the required depth of understanding to guide architectural decisions.

![Disciplines and Subsystems](image)

**Figure 74: The Architect's Depth of Understanding**

**Heuristics Overview**

*Heuristic 101:* Develop a first product that can be used as a basis for the product family.

*Heuristic 102:* Define and detail the architecture in such a way that BBs can be developed according to a simple waterfall model.

*Heuristic 103:* The architecture document describes the architectural models such as the BB dependency model, aspect designs, concurrency design and deployability design. The architecture
Document should be structured in a way which minimises the impact of changes.

Heuristic 104: The BB documentation consists of at least three documents: its specification document, its design document and its code document.

Heuristic 105: Consider a deviation of the BB development process from the simple waterfall a quality problem of the architectural process.

Heuristic 106: Proceed with the process of integration by extending a stable set of BBs with one or a few BBs.
Component-based development is one of the major trends in the software industry today. Many development projects base their products on component technology. Much attention is at present being given to the pros and cons of different component models. However, methods for identifying components are lacking [Szy98]. We regard our work as a contribution in the search for component methods.

The BBM is a component-based architectural design method for large software-intensive product families. It uses components to develop families of products in such a way that particular products can be configured from pre-manufactured components, that is, the BBs. The development of a family provides the context where BBs are identified and developed for multiple use in different products.

The architecting context of the BBM is a rational architecting process consisting of the tasks customer business modelling, application domain modelling, commercial product design, architectural design and technology. The tasks are described in their logical order, their execution is concurrent. The BBM supports the architectural design task of that model (see section 2.6).

Conceptual integrity is essential for the evolution of large systems. Because of the overwhelming amount of detail relevant in the development of such systems, architects get easily distracted from pursuing this integrity. The BBM presents a frame of reference for architectural design. Relevant design tasks and their underlying technical concepts are described (see chapter 3).

The BBM addresses the gap between domain functionality and system functionality by using system qualities and technology choices in the identification of objects and aspects (see chapter 5).

The core of the BBM consists of a number of design tasks, namely object design, aspect design, concurrency design, composability design and deployability design. It uses input from application domain modelling and commercial design (see chapter 3).