Architecture Modeling of Distributed and Concurrent Software Systems

Peter Klein, Diss. RWTH Aachen
Aachener Berichte zur Informatik Vol 31, Verlag Mainz, Aachen, 2000

This dissertation is devoted to the topic of *architecture modelling* for *software systems*. The architecture describes the structural composition of a system from components and relationships between these components. Thereby, it provides a basis for the system's realization on technical as well as on organizational level.

In order to define a conceptual frame and a suitable vocabulary for this working area, we present the *architecture* description *language* MoDeL (Modular Design Language). By selecting and combining modelling concepts which proved to be helpful for the design of software systems, this approach is integrative and pragmatic: It unites different realization paradigms (structured, object-oriented), different levels of granularity (overall system, subsystem, modules, types and operations), different kinds of abstractions (functional/data, types/objects etc.) without falling into a loose collection of diagram languages.

Considering an architecture as a construction plan in a cooperative, creative, and dynamic software development process, the necessity arises to allow flexible adaptations with respect to given constraints, e.g. performance, interaction with legacy systems, integration of external (sub-)systems, distribution in a network etc. On one hand, a clean structure is required to make the architecture in itself and its parts understandable, robust, maintainable, and reusable. On the other hand, it has to be dealt with that a realistic software development process may need divergence from the desired structure. A fundamental notion in this dissertation therefore is to consider a clean *logical architecture* as well as *concrete architectures* reflecting respective modifications as individual results of architecture modelling. Furthermore, transformation steps describing the changes induced by a particular realization constraint contain valuable modelling knowledge as well. If possible, this knowledge should be formalized to make it reusable.

A second basic aspect results from the proposition to separate the working areas of architecture modelling and of implementation, i.e. not to anticipate the steps during component development (selection of data structures, algorithms etc.) on architecture level. However, this cannot be avoided altogether as an architecture constrains the choice of possible realizations, e.g. by supplying certain underlying services. In order to allow the architect precise explanations of the static structure without giving up the distinction between architecture and realization, our approach supports the *annotation* of *dynamic runtime behavior* as well as *static component semantics* within certain limits.

In sum, we distinguish a separation in two dimensions, namely logical and concrete level on one hand and static and dynamic level on the other. These are described by respective sublanguages of the MoDeL language. From the broad range of adaptation steps towards a concrete architecture, we particularly focus on the *specification* of *concurrency* and *distribution*. This requires some new language concepts for fixing control flows, synchronization schemes etc. in the architecture.

Apart from the presentation of MoDeL and its sublanguages, this dissertation contains two *case studies*. These demonstrate the use of the language in different situations, but they also contribute to the discussion of "reference architectures".

Firstly, we present the detailed architecture of a *software development environment* which was developed in connection with the MoDeL language. This environment, the Analysis and Development Tool adt, is based on the experiences of the IPSEN project and supports the creation of MoDeL specifications. These specifications can be kept consistent with the source

text by fine-grained incremental propagation of changes from the architecture specification into source code files and vice versa. Additionally, the environment is open for the embedding of external tools, noticeably for programming activities (editor, compiler, debugger etc.). This facilitates an architecture-centered software development process by supporting design and realization under a consistent structural view.

The second architecture was developed in the course of the collaborative research center 476 “IMPROVE”. This project aims at the improvement of the information technology support of development processes in the application area of *process engineering*. A prototype of an *integrated development environment* has been implemented which extends existing development tools with new functionality, thereby enhancing single development steps as well as coordination and cooperation in the overall development process.

While the adt prototype allows for a detailed *discussion* of *logical architecture* concepts, the focus of the CRC prototype is the general design under consideration of constraints. Particularly, the *a-posteriori integration* of existing development tools for process engineering by new information technology concepts is discussed.

Referee: Prof. Dr.-Ing. M. Nagl
Coreferee: Prof. Dr. rer. nat. C. Lewerentz
Oral exam: July 17, 2000