The METAFrame'95 Environment

Bernhard Steffen, Tiziana Margaria, Andreas Claßen, Volker Braun

Lehrstuhl für Programmiersysteme
Universität Passau
D-94030 Passau (Germany)
{steffen,tiziana,classen,deepthou}@fmi.uni-passau.de

1 The METAFrameEnvironment

METAFrame is a *meta-level framework* designed to offer a sophisticated support for the systematic and structured computer aided generation of application-specific complex objects from collections of reusable components. Figure 1 shows its overall organization. Special care has been taken in the design of an adequate, almost natural-language specification language, of a user-friendly graphical interface, of a hypertext based navigation tool, and a of semi-automatic synthesis process and repository management. This application-independent core is complemented by application-specific libraries of components, which constitute the objects of the synthesis. The principle of separating the component implementation from its description is systematically enforced: for each application we have a distinct Meta-Data repository containing a logic view of the components. The tools themselves and their documentation are available in a different repository. This organization offers a maximum of flexibility since the synthesis core is independent of the direct physical availability of the tools (except for the execution, which is a different matter).

METAFrame constitutes a sophisticated programming environment for large to huge grain programs whose implementation is supported by the automatic, library-based synthesis of linear compositions of modules. More complex control structures glueing the linear portions together must be programmed by hand. Still, being able to synthesize linear program fragments drastically improves over other methods where only single components can be retrieved from the underlying repository. This is already true in cases where one is only interested in the functionality of single components, because our synthesis algorithm will automatically determine the required interfacing modules. Thus METAFrame supports the rapid and reliable realization of efficient application specific complex systems without sophisticated user interaction, making it an ideal means for a systematic investigation and construction of adequate implementations in a problem specific scenario.

The METAFrame approach abstracts from implementational details by allowing designers a *high-level-development* of the tools. *Specifications* express constraints in a *temporal logic* that uniformly and elegantly captures an abstract view of the repository. *Implementations* are in a *high-level language* tailored to express the combination of reusable analysis, verification and transformation components stored in the repository, which are considered as atomic on this level.

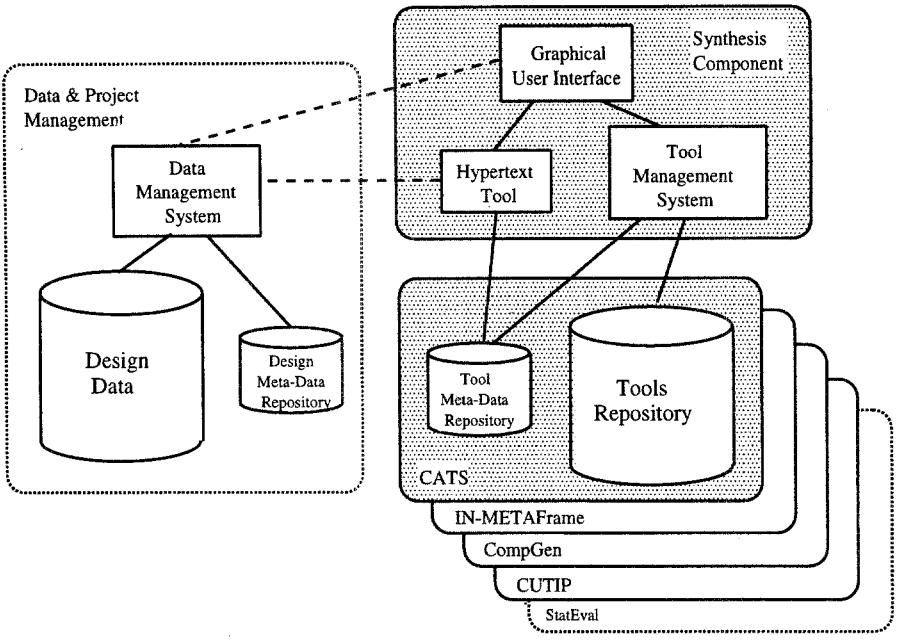

Fig. 1. The METAFrame Global Architecture

Synthesized systems are hierarchically structured. Since only the meta-structure, combining such components, is automatically generated from the specifications, the efficiency of the resulting system depends on the efficiency of the single components. In particular, different components may be written in different application languages (C, ML, C++) of different programming paradigms (imperative, functional, object-oriented).

METAFrame is currently available for UNIX systems, e.g. the Siemens Nixdorf RM line, and for PCs under LINUX. Its graphical interface as well as the hypertext browser are built on top of the Tcl/Tk graphics library [5]. Target language is the HLL, whose interpreter is implemented in C++.

2 Applications

The current experience has shown the practicability and flexibility of the METAFrame approach in several areas of application, which we briefly sketched in the following.

CATS. This application concerns the **C***omputer-***A***ided* **T***ool* **S***ynthesis* via automatic composition of heterogeneous verification algorithms for hardware and software systems from basic transformation and analysis components [4, 10]. It

supports the automatic generation of complex tools from a repository of components including equivalence checkers, model checkers, theorem provers, and a variety of decision procedures for application-specific problems. The number of basic components and infrastructure modules is steadily growing, as e.g. most of the tools come in many variants coping with infinite state-systems, values, time (discrete or dense), priorities, probabilities and combinations thereof, which all require their specific extensions of the 'basic' data structures.

IN-METAFrame. This project concerns the semi-automatic programming of *Intelligent Network Services* in cooperation with Siemens Nixdorf, Munich [7, 8]. IN-METAFrame is a development (creation) environment for reliable custom configurations of telephone services from a library of existing service-independent modules. The creation process is structured as follows: Initially, an existing service of similar application profile is loaded from the service library, or a completely new design from scratch is done (under model checking control). Alternatively, initial prototypes could automatically be generated from the set of underlying consistency conditions and constraints, a feature, which is not part of the current version of our service creation environment. These prototypes serve as a starting point for modifications that eventually lead to the intended service. Modification is guided by abstract views, and controlled by on-line verification of certain consistency constraints, guaranteeing the executability and testability of the current prototype.

CompGen. One of the major problems in data flow analysis or program optimization is the determination of the correct ordering of the individual analyses and optimizing transformations. Using METAFrame we can automatically syntesize complex heterogeneous optimization tools that respect certain important ordering constraints, guaranteeing correctness and often optimality of the overall algorithms [2, 3]. Moreover, in cases where there is not yet consensus about sensible orderings, METAFrame supports the corresponding investigation through its rapid prototyping facility. One should note here that the number of individual algorithms, many of which can be automatically generated (see[9]), may, like in the CATS-case, grow very large.

3 Evaluation and Perspectives

Our approach exactly meets the demands recently expressed by Goguen and Luqi in [1] for the emerging paradigm of *Domain Specific Formal Methods*: the essence of their proposal is to use formal methods on a large or huge grain level rather than on elementary statements, thus to support the programming with whole subroutines and modules as the elementary building blocks. This is precisely what METAFrame is designed for (see [6]).

Moreover, the possibility of natural language-like specification, the hypertext support of repository navigation, the specific profiles of the stored tools, and the user friendly graphical interface encourage successful experimentation without

requiring much expertise. In fact, one can use METAFrame also as an educational tool to train newcomers in an application field[1].

In the meantime, applications of METAFrame have been presented at various international fairs, like the CeBit'96 in Hannover, and, more interestingly, by Siemens Nixdorf with success at the TELECOM'95 in Geneva. GA*IN* (Global Advanced Intelligent Network), the IN solution jointly developed by Siemens and Siemens Nixdorf which includes IN-METAFrame, reached the market early this year and it has already been delivered to a number of customers, like e.g. Deutsche Telekom.

References

1. J.A. Goguen, Luqi: "*Formal Methods and Social Context in Software Development*," (inv. talk) TAPSOFT'95, Aarhus (DK), May 1995, LNCS N.915, pp.62-81.
2. M. Klein, J. Knoop, D. Koschützki, B. Steffen: "DFA&OPT-METAFrame*: A Tool Kit for Program Analysis and Optimization*" (Tool description) - Proc. TACAS'96, Int. Workshop on Tools and Algorithms for the Construction and Analysis of Systems, Passau, March 1996, LNCS 1055, Springer Verlag, pp. 422-26.
3. J. Knoop, O. Rüthing, B. Steffen: '*The power of assignment motion*," Proc. PLDI'95, ACM SIGPLAN, La Jolla, CA, June'95, SIGPLAN Notices 30, 6 (1995), pp.233-245.
4. T. Margaria, A. Claßen, B. Steffen: *"Computer Aided Tool Synthesis in the* METAFrame *"*, 3. GI/ITG Workshop on "Anwendung formaler Methoden beim Entwurf von Hardwaresystemen", Passau (D), March 1995, pp. 11-20, Shaker Verlag.
5. John K. Ousterhout: "*Tcl and the Tk Toolkit*," Addison–Wesley, April 1994.
6. B. Steffen, T. Margaria, A. Claßen, V. Braun: "*Incremental Formalization: a Key to Industrial Success* ", to appear in "SOFTWARE: Concepts and Tools", Springer Verlag, 1996.
7. B. Steffen, T. Margaria, A. Claßen, V. Braun, M. Reitenspieß: "*An Environment for the Creation of Intelligent Network Services*", invited contribution to the book "Intelligent Networks: IN/AIN Technologies, Operations, Services, and Applications – A Comprehensive Report" Int. Engineering Consortium, Chicago IL, 1996, pp. 287-300 – also invited to the *Annual Review of Communications*, IEC, 1996.
8. B. Steffen, T. Margaria, A. Claßen, V. Braun, V. Kriete, M. Reitenspieß: "*A Constraint-Oriented Service Creation Environment*" (Tool description) - Proc. TACAS'96, Int. Workshop on Tools and Algorithms for the Construction and Analysis of Systems, Passau, March 1996, LNCS 1055, Springer Verlag, pp. 418-421.
9. B. Steffen: "*Generating Data Flow Analysis Algorithms from Modal Specifications*," Science of Computer Programming N.21, 1993, pp.115-139.
10. B. Steffen, T. Margaria, A. Claßen: "*Heterogeneous Analysis and Verification for Distributed Systems*," "SOFTWARE: Concepts and Tools" N. 17, pp. 13-25, March 1996, Springer Verlag.

[1] At the University of Aarhus (Denmark) as well as in Passau it is used as a support in regular courses covering software techniques and formal methods in system development.