Generic Modelling with Graph Rewriting Systems

Manfred Münch, Diss. RWTH Aachen

One of the main goals of the software engineering discipline is the provision of appropriate tools for the construction of complex and interactive software systems. A large number of tools, languages, and methods exist to cope with the problems of administering the documents, information, and processes of creating large software systems. Specification languages and methods have been developed to describe formal aspects of these complex and interactive systems.

At the Department of Computer Science III at Aachen University of Technology, the specification language PROGRES (PROgrammed Graph REwriting System) has been developed in the context of the IPSEN (Interactive/Integrated/Incremental Project Support ENvironment) research project. With PROGRES it is possible to formally describe complex graph-based structures and operations on these structures. Furthermore, it is possible to generate rapid prototypes of the specified software systems as stand-alone applications.

In this thesis we have used the PROGRES specification system together with the rapid prototyping framework to build tools for editing, analysing, and interpreting visual languages. During a joint research with the Department for Process Control Engineering we have built such tools for an IEC-61131/3 compliant language which is widely used in the automation and process control industry. We have detected shortcomings of the PROGRES language for the specification of such large systems.

The identified problems of the PROGRES language are mainly dealing with reusability of specified code. Many modern programming languages offer genericity for modelling reusable software units, often together with an object-oriented programming methodology which also gives users support in an easy adaption of real-world problems to formal models. In this Thesis we have examined those programming and modelling languages and improved the PROGRES language by corresponding concepts. As being a statically typed specification language, PROGRES already has an elaborate two-level typing system. We have extended this typing system to allow for generic modelling of specification units.

The PROGRES graph model revealed another drawback for the specification of visual language systems during our research. It was only possible to specify the structure of a software system as an attributed graph and (graph) transformations on this structure. However, we were not able to define any behaviour of graph elements (nodes). Since a graph-based modelling approach already suggests an object-oriented specification methodology, the lack of transformation methods as part of graph elements became an obstacle for the specification of an extendible and generic visual language programming environment. Therefore, we have changed the graph model which PROGRES is based upon such that it is possible to assign a behaviour to graph elements. Furthermore, the effects of these changes on the rapid prototyping framework are discussed.

The PROGRES language and environment is very often used for rapid prototyping purposes. A very elaborate and flexible Java-based rapid prototyping environment UPGRADE was being developed at the Department of Computer Science III. In many collaborative research projects, as e.g. the IMPROVE project, we work with PROGRES and UPGRADE. Therefore, it was an important issue to ensure platform independence of the specified software systems. This is the reason why the prototype environment has been implemented in Java. In this thesis

we describe the tighter integration of PROGRES specifications and the prototyping environment by the generation of Java code from a specification. The final step for gaining a complete platform independence is the adaptation of the underlying database system on which PROGRES and UPGRADE are based. This is out of scope of this thesis, though.

Referee: Prof. Dr.-Ing. Manfred Nagl
Coreferee: Prof. Dr. Andreas Schürr
Oral exam: November 15, 2002