Electronic Communications of the EASST
Volume 30 (2010)

International Colloquium on Graph and Model Transformation - On the occasion of the 65th birthday of Hartmut Ehrig
(GraMoT 2010)

From Separate Formal Specifications to Certified Integrated Visual Modelling Techniques and Environments (Position Statement)

Hartmut Ehrig

7 pages

Guest Editors: Claudia Ermel, Hartmut Ehrig, Fernando Orejas, Gabriele Taentzer
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

From Separate Formal Specifications to Certified Integrated Visual Modelling Techniques and Environments (Position Statement)

Hartmut Ehrig

ehrig@cs.tu-berlin.de
TFS-Group
Technische Universität Berlin

Abstract: In this position statement we discuss the state of the art and role of formal specification and modelling techniques in different periods with special focus on the work of the TFS-group at TU-Berlin. In the past (1970 – 1990) single formal specification techniques have been developed with little impact on practical software development. In the present (1990 – 2010) integrated and visual modelling techniques have gained more and more importance. For the future (2010 – 2020) we try to sketch the idea of a *Certified Integrated Visual Modelling Technique and Environment* based on an integration of graph theory, graph transformation and Petri net theory, short *Dynamic Graph and Net Theory*.

Keywords: Software system modelling**.** formal specification, modelling techniques, visual modelling , TFS – group

1 Introduction

This paper is a position statement for the panel discussion at the International Colloquium on Graph and Model Transformation at TU-Berlin, February 11-12, 2010. Past, presence and future of formal aspects of system modelling with special focus on the work of the TFS (Theoretische Informatik : Formale Spezifikation) –group at TU-Berlin are discussed in the following three sections :

- Formal Software Specification Techniques in the Past (1970 – 1990)
- Formal Software System Modelling in the Present (1990 – 2010)
- Future Aspects of Formal Software System Modelling (2010 – 2020)

2 Formal Software Specification Techniques in the Past (1970 – 1990)

In the beginning software development was just programming of algorithms and data types, especially implementation of numerical algorithms in FORTRAN and ALGOL. This means mathematical modelling in programming was mainly numerical mathematics, although mathematical notions of algorithms had been developed already in mathematics based on Turing machines, recursive functions, and λ-calculus. These ideas were picked up in theoretical computer science leading to the areas of algorithms and complexity theory.

The concept of abstract data types – on the other hand – was developed in the early 1970ies by computer scientists like Parnas and Hoare influenced by the debacles of large software systems in the late 1960ies. At that time it was not at all clear whether abstract data types and software systems in general could be modeled by mathematical concepts. On the contrary, the vast majority of practical computer scientists at that time was convinced that mathematical methods may be useful for numerical analysis of algorithms, but not at all for software system development.

In addition to automata theory, formal languages, algorithms, and complexity the main challenge for theoretical computer science was to find mathematical models for programming language semantics, abstract data types, and concurrency of processes. *The ultimate goal was to define correctness of software systems w.r.t. given formal requirements and to develop compositional correctness and proof techniques*.

First steps in this direction in the 1970ies were the development of operational and denotational semantics of programming languages by pioneers like Scott, Stratchey,de Bakker and Nivat. The concepts of Petri nets and process calculi have been defined by other pioneers like Petri, Milner, Hoare, Rozenberg, and Montanari in order to model concurrent and distributed processes.

The main contributions of the TFS-group in this period were :

1. A categorical approach to automata theory in order to unify different kinds of deterministic, partial, stochastic, nondeterministic, and topological automata [EKKK74].
2. Development of the double pushout (DPO) approach for graph transformation – together with Schneider and Rosen - in order to define graph languages and to model operational semantics [Ehr79].
3. The fundamentals of algebraic specification of data types and software modules in the initial algebra approach [EM85/90], based on the pioneering work of Goguen, Thatcher, Wagner, and Wright [ADJ75].

In order to show how theory and practice of software development can influence each other the TFS- and SWT-group at TU-Berlin, chaired by H. Ehrig and Ch. Floyd respectively, created and organized together with M. Nivat (Paris) and J. Thatcher

(Yorktown Heights, USA) the new conference series TAPSOFT (Theory and Practice of Software Development). It was held with great success at TU-Berlin in 1985 bringing together several pioneers from theoretical and practical computer science and about 500 researchers for both areas. TAPSOFT was continued successfully as biannual conference until 1997 at different prominent locations all over Europe. It certainly helped to bridge the gap between theory and practice in software development.

Summarizing, the first steps of formal software system specification in the past have been difficult, but at least several promising formal specification techniques for data types, modules, and concurrent processes have been developed.

3 Formal Software System Modelling in the Present (1990 – 2010)

Starting in the 1990ies formal methods, especially formal specification techniques, were supported by several basic research actions and projects launched by the European Community (EC). The TFS-group was especially involved in COMPASS, on algebraic specification techniques, and in COMPUGRAPH, GETGRATS, APPLIGRAPH, and SEGRAVIS on graph transformation techniques.

Graph grammars and transformation systems had been developed in the past mainly from the language and the process specification point of view [Roz97], but data types had to be handled separately by algebraic specification techniques. This situation was typical in the past and there was a need for integrated techniques simultaneously taking care of data types and processes for software systems. Typical examples of integrated specification techniques studied by the TFS-group have been LOTOS, High-Level Nets, and Attributed Graph Grammars (AGG) integrating algebraic specifications with CCS, Petri nets, and graph transformation systems respectively.

This need for integration was supported by the priority program SoftSpez (Integration of Software Specification Techniques for Applications in Engineering) of the German Research Council (DFG). This was initiated by W. Brauer, M. Broy, H. Ehrig (coordinator), H.-J. Kreowski, H. Reichel and H. Weber from computer science, and E. Schnieder and E.Westkämper from engineering [Ehr et al 2004].

But the need for integration was not only supported by integrated techniques, but also by the integration of different views, most prominently supported by UML [UML 2.0]. Today UML is an international standard for object oriented software development and also a well-known visual modelling technique, which is widely used in practice. The semantics of UML, however, is mainly informal and various attempts have been made to provide a formal semantics for specific UML diagram techniques. In the last decade it turned out that typed attributed graph transformation is an intuitive and powerful visual modelling technique. Moreover it has a precise formal semantics and a rich mathematical theory [EEPT06], which supports correctness for visual modelling of software systems. In addition to visual modelling also model transformation becomes more and more important for efficient construction, correctness and consistency of software system modelling. In fact, graph transformation techniques are also most suitable for model transformations between visual and visual or textual modelling languages. In fact, model transformations can be seen as a generalization of compilation between textual languages. Syntactical and semantical correctness of model transformation can be supported by the theory of algebraic graph transformation [EEPT06] and simulation and analysis techniques by the tool AGG [AGG] developed by the TFS-group. Especially the concept of triple graph grammars introduced by A. Schürr [Sch94] is very useful for model transformation and integration and a promising formal construction and semantics developed in [EEHP09].

A most prominent forum, where the results of formal software system modelling have been presented, are the ETAPS conferences. The European Joint Conferences on Theory and Practice of Software (ETAPS) were created in 1998 as annual conferences, combining especially the well-established biannual conferences TAPSOFT and ESOP. After the first ETAPS conferences in Lisbon 1998 and Amsterdam 1999 ETAPS was continued with great success at TU Berlin organized by H. Ehrig, S. Jähnichen, B. Mahr (general chair) and P. Pepper, based on the good experience with TAPSOFT in 1985. Today ETAPS is the most prominent joint conference combining theory and practice of software in Europe, supported by the European associations EATCS, EAPLS, and EASST. Moreover, ETAPS is well-established on the international level. Especially graph transformations were supported by the international graph grammar workshops from 1978 until 1998 and the international conference on graph transformation (ICGT) since 2002 in Barcelona, Rome, Natal, Leicester, and coming up in Twente and Bremen.

Summarizing the situation today, formal software system modelling – at least for small systems - is well-accepted in theory and practice, especially visual model-ling using graph transformation techniques. Software system modelling is mainly based on integration and visualization of models with model transformations supported by various construction, analysis, and verification techniques.

4 Future Aspects of Formal Software System Modelling (2010 – 2020)

The ultimate goals for the semantical challenges in the past are mainly valid today, but we have done important steps to realize them at least for small systems already. Let us rephrase the goals for the future as follows :

Formal software system modelling should support the modular development and integration of correct software systems in the large, where construction, correction and verification are based on visual models and model transformations using compositional semantics and proof techniques.

In section 3 we have discussed that some of these aspects are realized for small systems already today. Of course, it is important to make sure that these techniques scale up for large systems. But what is missing in addition to realize the ultimate goals in the future? Let us point out the following two aspects concerning theory and practice of graph transformation systems :

1. Dynamic Graph and Net Theory
2. Certified Integrated Visual Modelling Techniques and Environments

The idea of dynamic graph and net theory is an integration of mathematical graph theory and optimization of algorithms in the sense of the MATHEON research center at TU-Berlin with the theory of graph transformations and Petri nets in theoretical computer science. Part of this integration has been done by the TFS-group leading to the concept of reconfigurable and higher-order Petri nets. This allows a dynamic interaction of rule based modification of the Petri net structure with the well-known token game [PEHP08] on one hand and nets and rules as token on the other hand. However, the interaction of graph algorithms with rule based transformations of the underlying graphs is certainly promising for the evolution of algorithms in a changing environment, but only little work has been done in this direction up to now.

The idea of a certified integrated visual modelling technique and environment is an integration of visual modelling techniques inspired by UML with advanced graph and model transformation techniques. These techniques should be supported by powerful analysis, model checking, and theorem proving techniques. First steps for such an integration is first of all the PhD-thesis of L. Lambers [Lam09] concerning certification of rule based modelling supported by the AGG-system [AGG]. Other important steps are efficient and correct model transformations based on triple graph grammars [EEHP09], and the work of A. Habel and K.-H. Pennemann [HP08] on correctness and verification of nested graph constraints for graph transformations.

References

[ADJ75] ADJ-Group (Goguen, Thatcher, Wagner, and Wright): Abstract Data Types as Initial Algebras and the Correctness of Data Representation.Proc. Conf. on Computer Graphics, Pattern Recognition and Data Structures (1975)

[AGG] Attributed Graph Grammar Tool AGG : http://tfs.cs.tu-berlin.de/agg/ (2010)

[Ehr79] Ehrig, H. : Introduction to the Algebraic Theory of Graph Grammars (A Survey), Springer LNCS 73 (1979), 1- 69

[Ehr et al 04] Ehrig, H. et al. (Editors) : Integration of Software Specification Techniques for Applications in Engineering. Final Report DFG Priority Program SoftSpez, Springer LNCS 3147 (2004)

[EEHP09] Ehrig, H.,Ermel, C.,Hermann, F., Prange, U.: On-the-Fly Construction, Correctness and Completeness of Model Transformations based on Triple Graph Grammars, Proc. MODELS'09, (2009)

[EEPT06] Ehrig, H., Ehrig, K., Prange, U., Taentzer, G. : Fundamentals of Algebraic Graph Transformation, EATCS Monographs, Springer 2006

[EKKK74] Ehrig, H., Kiermeier, K.-D., Kreowski, H.-J., Kuehnel, W. : Universal Theory of Automata, Teubner (1974)

[EM85/90] Ehrig, H., Mahr, B. : Fundamentals of Algebraic Specification 1 and 2, EATCS Monographs, Springer (1985/1990)

[HP08] Habel, A., Pennemann, K.-H. : Correctness of High-Level Transformation Systems Relative to Nested Conditions, MSCS 19 (2008), 245-296

[Lam09] Lambers, L. : Certification of Rule Based Modelling by Graph Transformation, PhD-thesis, TU-Berlin (2009)

[PEHP08] Prange, U.,Ehrig,H.,Hoffmann,K.,Padberg,J. : Transformations in Reconfigurable Place/Transition Nets, Springer LNCS 5065 (2008), 96-113

[Roz97] Rozenberg, G. (Editor) : Handbook of Graph Grammars and Computing by Graph Transformation.Vol.1 Foundations, World Scientific (1997)

[UML2.0] Unified Modelling Language UML 2.0 : http://www.omg.org/uml/ (2003)