Metamodel-based Integration of Tools

Peter Braun*
Institut für Informatik
Technische Universität München
D-85748 Garching b. München, Germany

Abstract: This paper will sketch some results of current research within the projects AutoFOCUS/Quest and Automotive. AutoFOCUS/ Quest is a prototype for a development tool showing how model-based techniques could be used to deal with the complexity of embedded software development. Within Automotive the integration of commercial tools is examined. Therefore a common conceptual metamodel is developed which defines the core of the Automotive Modeling Language and is the base of the integration.

Keywords: metamodel, tool integration, model-based development, automotive software development

1 Introduction

About thirty years ago the first electronic control units (ECUs) were used within cars. In the beginning those ECUs controlled the fuel injection of engines. The aim was to minimize fuel consumption and simultaneously increase the engine's power. About ten years later the first programmable devices were deployed. They implemented the first Anti-Lock Braking System (ABS). Nowadays up to 80 ECUs are used within premium class cars. They are responsible for a number of different tasks ranging from the motor management over driver assistance to different comfort functions. These systems are not only more and more important for a successful competition but they are also necessary to ensure the unendangered operation of a car.

So the development of the ECUs especially with safety relevant functions has to be done conforming to standards like [18, 23]. As the different ECUs are connected and have to exchange data for their operation, the development of ECUs is a challenging task. Motivated by other engineering disciplines where models are necessary to deal with the inherent complexity of the environment by reducing it to relevant details, also within software engineering model-based processes promise to improve the traceable construction of the software parts of ECUs.

Current CASE tools especially for the development of embedded systems support only parts of the complete development process. So there exists a variety of commercially available and specialized tools for different tasks. Examples are tools like DOORS [37], for the management of requirements, UML Suite [38], for building abstract models of developed systems, and ASCET-SD [13], for the fine design and code generation. For a consistent and cost efficient development process those tools have to be integrated. This leads to two important problems which have to be solved:

- First an integrated method and a language has to be constructed. Thereby the language and the method are heavily influenced by the chosen tools.
- Second the integration of the tools has to be specified and implemented.

Within the FORSOFT project Automotive the above mentioned tools are integrated. Therefore the Automotive Modeling Language (AML) [40, 39] is defined by a metamodel specifying the concepts, their relations, and some constraints. The metamodel defines the abstract syntax of the AML. For the representation of models the concrete syntax of the involved tools is used. Thereby the languages defined by the tools are restricted to subsets.

The prototypic integration of the tools is specified based on the AML metamodel and the metamodels of the tools. Therefore the Bidirectional Object-oriented Transformation Language (BOTL) [4, 5] is used. BOTL allows the specification of transformation rules which define how parts of a source model are transformed to parts of a destination model. As the realization of the prototype of the Automotive tool chain is done by the tool vendors ETAS and Telelogic, this specification is based upon the UML so that it is easily understood by programmers.

In the following some key factors of the model-based software development and the metamodel-based integration are sketched briefly.

*Email: braunpe@in.tum.de

2 Model-based Software Development

Nowaday model-based specification techniques are becoming more and more popular. This popularity results among other from the UML [27] which is defined by a metamodel. A *(conceptual) metamodel* defines the entities and relations which are instantiated within a *(conceptual) model* that describes a developed system. A metamodel also defines some *consistency constraints*. The *semantical model* defines the meaning of the entities and relations of the conceptual model.

In our opinion model-based development is characterized by the following ingredients [35]:

Explicit (meta-)models: A conceptual metamodel describes the abstract syntax of the used notations. It also defines invariant constraints which have to hold for all models. So the possible products of a development process are defined systematically by this conceptual metamodel.

Precise foundation: Not only the meaning of a metamodel but also the meaning of developed models has to be defined precisely. So a formally founded semantical base is needed.

Tool support: Tools based upon the conceptual metamodel and the semantic foundation are necessary for the efficient use of model-based methods. Tools restrict the development process in a reasonable way and allow a purposeful process.

Especially a precisely founded semantics is often missing in current commercially available tools. In the example of UML this leads to system descriptions in different views for which the integration into a consistent, realizable model is hard.

Also for a formal-based validation of properties of a developed system a precisely defined semantic is necessary. This enables the use of formal verification techniques like model checking or theorem proving, or enhanced testing methods as this is done in AutoFOCUS/Quest [17, 6, 32, 3].

3 Metamodel-based Integration

Tool support is an absolute necessity for practical development processes in the automotive domain. Thereby commercially available tools have to be used. As the development of the software parts of ECUs with "traditional" methods is becoming more and more critical, model-based techniques gain importance. An example for this is the research project Automotive where it is the aim to develop an automotive specific development method for embedded software and demonstrate this by the integration of commercial tools.

A main part of this project was the definition of a metamodel used to describe the static structure of embedded software. This metamodel defines the language AML which is a specialized architecture description language [39]. A key feature of the AML is the definition of variants of system parts. Variants of systems play an important role within the automotive industry.

Based upon the metamodel and the metamodels of the tools DOORS, UML Suite, and ASCET-SD the integration is defined. Thereby abstract conceptual metamodels for ASCET-SD and DOORS had to be reconstructed from their object-oriented data models. As already mentioned the implementation of the tool chain is done by the tool vendors themselves. This is not an unusual situation within the automotive industry. Thereby often a company specific method is defined respectively adapted by some experts and implemented externally.

So the integration of the tools has to be specified in an abstract way, preferably in a notation to which programmers and method designers feel comfortable. Thus in the project Automotive a UML-based specification language for a metamodel-based integration was developed. This Bidirectional Object-oriented Transformation Language (BOTL) [5] is used to specify the transformation of models of the different tools to a common "exchange format" defined by the AML metamodel.

A BOTL specification consists of a *rule set* describing rules which define the transformation of parts of a source model to parts of a destination model (cf. Fig. 1). Thereby both sides of a rule are defined by so called *model variables* which are similar to object diagrams but use terms instead of concrete values for attributes and identifiers. The source and target metamodels are defined by class diagrams.

In Figure 1 an example rule is shown which relates a part of the AML metamodel to a part of the ASCET-SD metamodel. It specifies that a pattern consisting of object variables specified on the left hand side are transformed to two objects specified by the right hand side and vice versa. Objects have unique identifiers shown before the objects type. These may be so-called primary key attributes shown in bold face. Each rule defines a equational system which must have at most one solution. This equational system defines how attribute values of one side are calculated by attribute values of the other side. So during a rule application occurrences of model fragments of the source side are searched and according model fragments of the target side are constructed and merged. An important, formally proven property of BOTL rules is that rule applications are commutative and associative.

BOTL not only allows to define rule sets but also defines properties of rule sets. A rule set is called *applicable*, if the transformation it defines can be applied

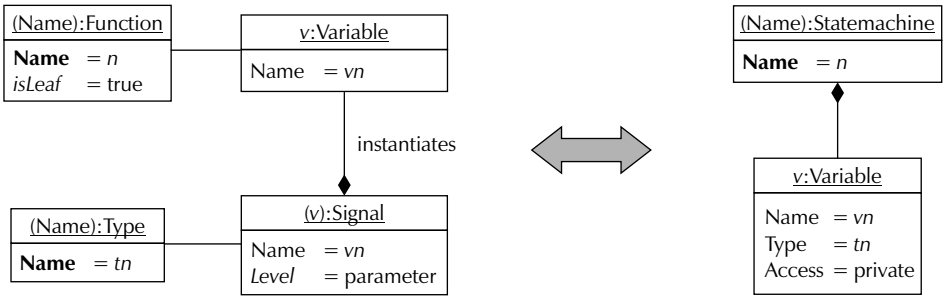

Figure 1: An example BOTL rule transforming variables

for any arbitrary source model. Thereby an applicable transformation is deterministic, that means it will produce always the same result for a given source model.

If it holds that all possibly created target models are conform to the target metamodel, then the according rule set is called *metamodel conform*.

Obviously both properties are crucial for the definition of transformation specifications. Thus BOTL is designed to prove these properties tool supported. The BOTL formalism enables to prove these properties based on the metamodels and a given rule set. So it is possible to ensure that a given rule set transforms every possible input model into reasonable output models.

Also the concrete transformation of models could be automated, although this is not helpful for the situation in Automotive, as the tools use their own proprietary data format. In a productive environment some kind of "online" translation of data between different tools is needed but this was not the focus of the research in Automotive.

Compared to other approaches BOTL enables the specification of bidirectional transformations, which may be processed automatically. The formal foundation of BOTL allows to argue about general properties of transformation specifications. So a rule set designer could be guided. As each rule defines an equational system which has at most one solution the possible transformations are restricted. For example non deterministic transformations are not possible. Also the specification of a negative context, specifying for example that an object is not connected to an other object, is not (directly) possible.

4 Related Work

The transformation of models is a key feature in many domains. Beside the generic approaches in the field of schema evolution [9, 8], mediator-based data integration [7], and federated database systems [21, 36] some specific approaches deal with the transformation of models in the CASE-supported development area. Three main techniques which are used for example in the area of Model Driven Architecture [20] are described in the following. The Model Driven Architecture approach itself focuses on the mapping between platform independent and platform dependent models but does not provide a formalism for describing model transformations [14]. Some major requirements for such a transformation language are [19]:

- bidirectional mappings should be possible,
- a bidirectional mapping should be specified within one definition,
- the mappings should be defined fine grained with a well defined semantic, and
- tool support for automatic mapping processing.

4.1 XML-based transformations

XSLT [11] provides a language for the specification of the transformation of XML-based documents. XMI [28] enables the representation of MOF-based models as XML-based documents. By combining both technologies model transformations could be specified. As the syntax of XML/XMI is very verbose also the transformation specifications in XSLT are usually huge and hard to read. It is also difficult to differentiate parts dealing with the source model from parts dealing with the target model. And even more badly an abstract notion of a model gets lost. Also practical applications have shown that it is not easy to track down specification errors. Unfortunately this leads to the fact that XSLT is hard to use for model transformations.

Thus some approaches try to hide XSLT [30, 31] by using a more abstract language. The xlinkit approach goes a different way [25, 26]. Based on XML technologies like XPath [10] and Xlink [12] xlinkit provides a mechanism for generating consistency links between XML-based documents. Consistency constraints are defined based on first order logic rules which are represented also in XML. By the application of the xlinkit

toolkit these constraints are checked and new links are generated for every model element satisfying a rule. The purpose of these links is to inform users about inconsistencies within their documents. As dissolving inconsistencies is usually not trivial the user has to correct his documents manually. Thereby the generated links help the user to identify model elements which have violated or fulfilled a specific constraint.

In a document oriented development process this approach seems to be appropriate. Within a model-based development process the models have to be translated into XML documents. Afterwards the consistency rules have to be defined based upon these documents types. For an ergonomic presentation of inconsistent model elements the XML representation of a model element has to be retranslated to the more abstract model oriented representation. The overall process for doing this is not straightforward.

4.2 UML/OCL-based transformations

Triggered by the need of transformation within MDA different approaches are arising. Some of them use the combination of UML resp. MOF and OCL for specifying transformations. More or less typical approaches are described in [15, 29]. A similar approach is described in more detail in [2, 1]. Typically the specification is split into two pieces. A graphical part displaying two class or object diagrams and an abstract definition of mappings between them, and a textual part specifying those mappings. Thus a mapping is defined abstractly with the help of enriched UML class or object diagrams. Two conceptual models or parts of them are shown side by side. Between them mapping relations are defined graphically specifying classes respectively objects of one model which are mapped onto classes respectively objects of the other model. Also these mapping relations could be categorized for example in unidirectional or bidirectional mappings. Further shortcuts for specific types of mappings could be used. The specification of each mapping is done textually by using OCL. Within these OCL constraints for example the equivalence of some attributes is described.

This technique allows the abstract definition of a mapping on a high level. So it is possible to show the relations between different parts of conceptual models. But normally most of the interesting details are "hidden" in OCL expressions. The usage of OCL implies that a efficient tool supported interpretation of such mappings is not always possible.

4.3 Graph transformations

Graph grammars and graph transformation systems [33] also enable the definition of model transformations based on rules. Examples for systems based on graph transformation are DIAPLAN [16] or DIAGEN [22]. Beside parsing an arbitrary edited graph according a graph grammar e.g. the definition of structured editing operations is possible. Those operations transform a graph representing a model into a new graph which represents again a consistent model. As a graph grammar is defined in rules it is not always easy to explicitly identify the constraints which hold for the specified graphs. This can be a drawback if one has to reason about a models defined by a graph grammar. However, since this approach focuses on genericity concerning the transformed diagrams, issues like the integration of different views of a complex instance model are not in the main focus.

Triple graph grammars [34] provide a mechanism how two different graphs described by graph grammars can be integrated by a third graph grammar. This mechanism can be used to integrate different views. In the IPSEN approach [24] several kinds of integration are described. Based upon the graph transformation system PROGRES a tightly integrated software development environment was sketched. This research work shows how far graph grammars can be used as for the description of a software development process and as an integration mechanism. But it enlightens also some drawbacks. For example in the so called horizontal integration simple (invariant conceptual) consistency constraints are defined by graph grammars, which are currently described more compact in a model-based approach with an explicit (conceptual) metamodel. However, graph grammars provide a theoretically sound approach which can help to solve practical integration problems.

5 Conclusion

In this paper we have briefly sketched some results of research in the projects AUTOFOCUS/Quest and Automotive. With AUTOFOCUS/Quest we have realized a model-based CASE tool for the development of embedded systems. We have integrated AUTOFOCUS/Quest with some formal verification tools to show how these could enrich development processes in the future.

Within Automotive we have defined an automotive specific specification language based upon commercially available and practically used tools. We have defined a rule-based transformation specification language which is used to define the integration of three tools.

Further work will go into two directions. First of all we are currently developing tool support for BOTL. Currently a editor for rule sets and a tool for verifying the properties of applicability and metamodel conformance are developed. Also a tool for generating code for the transformation of object oriented models

is planed.

Second, AutoFocus/Quest will be developed further. Especially the model-based support of development processes has to be examined in more detail.

References

[1] David H Akehurst. *Model Translation: A UML-based specification technique and active implementation approach*. PhD thesis, University of Kent at Canterbury, 2000.

[2] David H. Akehurst and Stuart J. H. Kent. A Relational Approach to Defining Transformations in a Metamodel. In Jean-Marc Jezequel and Heinrich Hussmann, editors, <> *2002 - The Unified Modeling Language: Model Engineeing, Concepts, and Tools*, volume 2460 of *Lecture notes in computer science*. Springer, October 2002.

[3] P. Braun, H. Lötzbeyer, B. Schätz, and O. Slotosch. Consistent Integration of Formal Methods. In *Proc. 6th Intl. Conf. on Tools for the Analysis of Correct Systems (TACAS), LNCS 2280*, 2000.

[4] Peter Braun and Frank Marschall. Transforming Object Oriented Models with BOTL. In Paolo Bottoni and Mark Minas, editors, *International Workshop on Graph Transformation and Visual Modeling Techniques*, number 72.3 in ENTCS. Elsevier Science B. V., 2002.

[5] Peter Braun and Frank Marschall. BOTL—The Bidirectional Objekt Oriented Transformation Language. Technical Report TUM-I0307, Fakultät für Informatik, Technische Universität München, 2003.

[6] Peter Braun and Oscar Slotosch. Development of a car seat: A case study using autofocus, doors, and the validas validator. In *OMER—Object-oriented Modeling of Embedded Real-Time Systems*, LNI P-5. Springer, 2002.

[7] S. Busse. A Specification Language for Model Correspondence Assertions. Technical report, Technische Universität Berlin, Fachbereich 13 Informatik, Computergestützte informationssysteme, 1999.

[8] M.R. Cagan. The HP SoftBench Environment: An Architecture for a New Generation of Software Tools. *Hewlett-Packard Journal*, pages 36–47, June 1990.

[9] E. Casais. *Managing Class Evolution in Object-Oriented Systems*, volume Object-Oriented Software Composition of *The Object-Oriented Series*, chapter 8, pages 201–244. Prentice Hall, 1995.

[10] J. Clark and S. DeRose. XML Path Language (XPath) Version 1.0. Recommendation, World Wide Web Consortium (W3C), Nov. 1999. http://www.w3.org/TR/1999/REC-xpath-19991116.

[11] James Clark. XSL Transformations (XSLT) Version 1.0. Technical report, World Wide Web Consortium (W3C), 1999. http://www.w3.org/TR/xslt.

[12] S. DeRose, E. Maler, and D. Orachard. XML Linking Language (Xlink) Version 1.0. Technical report, World Wide Web Consortium (W3C), June 2001. http://www.w3.org/TR/xlink.

[13] ETAS GmbH, Stuttgart. *ASCET-SD User's Guide Version 4.2*, 2001.

[14] Anna Gerber, Michael Lawley, Kerry Raymond, and Jim Steel. Transformation: The Missing Link of MDA. In Andrea Corradini, Hartmut Ehrig, Hans-Jörg Kreowski, and Grzegorz Rozenberg, editors, *Graph Transformation*, LNCS 2505, pages 90–105. Springer, 2002.

[15] JH Hausmann and S Kent. Visualizing Model Mappings in UML. In *Proceedings of the ACM Symposium on Software Visualization*, 2003. To appear.

[16] Berthold Hoffmann and Mark Minas. Towards Rule-Based Visual Programming of Generic Visual Systems. In *2002 ACM SIGPLAN Workshop on Rule-Based Programming*. ACM, 2002.

[17] Franz Huber and Bernhard Schätz. Integrated Development of Embedded Systems with AutoFocus. Technical Report TUM-I0107, Fakultät für Informatik, Technische Universität München, 2001. http://wwwbib.informatik.tu-muenchen.de/infberichte/2001/TUM-I0107.ps.gz.

[18] International Electrotechnical Commission. *Functional safety of electrical/electronic/ programmable electronic safety-related systems*, 1998. IEC 61508.

[19] S Kent and R Smith. The Bidirectional Mapping Problem. *Electronic Notes in Theoretical Computer Science*, 82(7), 2003. To appear.

[20] Stephen J. Mellor, Kendall Scott, Axel Uhl, and Dirk Weise. Model-driven architecture. In Jean-Michel Bruel and Zohra Bellahsène, editors, *Advances in Object-Oriented Information Systems*, LNCS 2426, pages 290–297. Springer, 2002.

[21] W. Meng and Y. Clement. Query Processing in Multidatabase Systems. In W. Kim, editor, *Modern Database Systems*. ACM Press, 1995.

[22] Mark Minas. *Spezifikation und Generierung graphischer Diagrammeditoren*. Habilitation, Universität Erlangen-Nürnberg, 2001.

[23] The Motor Industry Software Reliability Association (MISRA), Nuneaton, Warwickshire, UK. *Guidelines for the use of the C Language in Vehicle Based Software*, April 1998.

[24] M. Nagl, editor. *Building Tightly Integrated Software Development Environments: The IPSEN Approach*. Springer, 1996.

[25] C. Nentwich, L. Capra, W. Emmerich, and A. Finkelstein. xlinkit: A Consistency Checking and Smart Link Generation Service. *ACM Transactions on Internet Technology*, 2(2):151–185, May 2002.

[26] C. Nentwich, W. Emmerich, and A. Finkelstein. Better Living with xlinkit. In *Proc. 2nd International Workshop on Living With Inconsistency at ICSE 2001, Toronto*, May 2001.

[27] OMG. OMG Unified Modeling Language Specification. Technical Report 1.4, formal/01-09-67, Object Management Group (OMG), www.omg.org, 2002.

[28] OMG. OMG XML Metadata Interchange (XMI) Specification. Technical Report 1.2, formal/02-01-01, Object Management Group (OMG), http://www.omg.org, Jan. 2002.

[29] OMG. QVT Partners Initial submisison to the MOF 2.0 Q/V/T RFP. Technical Report ad/03-03-27, Object Management Group (OMG), http://www.omg.org, 2003.

[30] Mikaël Peltier, Jean Bézivin, and Gabriel Guillaume. MTRANS: A general framework, based on XSLT, for model transformations. In *Workshop on Transformations in UML (WTUML), Genova, Italy*, April 2001.

[31] Mikaël Peltier, Francois Ziserman, and Jean Bézivin. On levels of model transformation. *XML Europe 2000, Parice, France*, June 2000.

[32] J. Philipps and O. Slotosch. The Quest for Correct Systems: Model Checking of Diagramms and Datatypes. In *Asia Pacific Software Engineering Conference 1999*, pages 449–458, 1999.

[33] Grzegorz Rozenberg, editor. *Handbook of Graph Grammars and Computing by Graph Transformation*, volume 1: Foundations. World Scientific, 1997.

[34] A. Schürr. Specification of graph translators with triple graph grammars. In G. Tinhofer, editor, *WG'94 20th Int. Workshop on Graph-Theoretic Concepts in Computer Science, Herrsching, Germany*, LNCS 903, pages 151–163. Berlin: Springer Verlag, Juni 1994.

[35] B. Schätz, P. Braun, F. Huber, and A. Wisspeintner. Consistency in Model-Based Development. In *Tenth IEEE International Conference and Workshop on the Engineering of Computer-Based Systems*, pages 287–296. IEEE Computer Society, 2003.

[36] A.P. Sheth and J.A. Larson. Federated Database Systems for Managing Distributed, Heterogeneeous, and Autonomous Databases. In *ACM Computing Surveys*, number 22-3, pages 183–236. 1990.

[37] Telelogic AB. *Using DOORS*, 2001.

[38] Telelogic AB. *UML Suite 4.6, Getting Started*, 2002.

[39] Michael von der Beeck, Peter Braun, Ulrich Freund, and Martin Rappl. Architecture Centric Modeling of Automotive Control Software. In *SAE Technical Paper Series 2003-01-0856*, 2003.

[40] Michael von der Beeck, Peter Braun, Martin Rappl, and Christian Schröder. Automotive Software Development: A Model Based Approach. In *SAE Technical Paper Series 2002-01-0875*, 2002.