In: System Development Topsy Turvy − Controlling the Process. Proc. Annual Congress SBIT, Tilburg University, March 24, 1998.

Conceptual Modeling in a Computerised World

Manfred A. Jeusfeld

KUB Tilburg, Infolab
jeusfeld@kub.nl

The diversity of modeling purposes lets standard development tools appear less and less flexible. They force developers (including the users) learn some fixed set of notations. Traditionally, models are stepwisely transformed from the user side (requirements) to the implementation side (programs). As a result, the minds and skills of humans are restricted by the prescribed notations and languages. The language determines the thoughts. There must be a way out of this dilemma. We call it meta modeling!

Conceptual models offer abstract views on certain aspects of the real world (description role) and the information system to be implemented (prescription role) [Yourdon89]. They are used for different purposes, such as a communication medium betweenusers and developers, for managing and understanding the complexity within the application domain, and for making experiences reusable. The presence of multiple conceptual modeling languages is common in information systems engineering as well as other engineering disciplines. The reasons are among others:

the complexity of the system requires a decomposition of the modeling task into subtasks; a frequent strategy is to use orthogonal perspectives (data view, behavioral view, etc.) for this decomposition

the information system is decomposed into subsystems of different type, e.g. data storage system vs. user interface; experts for those subsystems tend to prefer special−purpose modeling languages

the modeling process is undertaken by a group of experts with different background and education; the experts may have different preferences on modeling languages

conceptual modeling has different goals (e.g., system analysis, system specification, documentation, training, decision support); heterogeneous goals lead to heterogeneous representation languages, and to heterogeneous ways−of−working even with given languages

The pre−dominant approach to solve the integration problem is to "buy" an integrated CASE tool which offers a collection of predefined modeling languagesand to apply it in the manner described in the manual. There are good reasons to do so: the method design has already been done and the interdependencies between the multiple modeling languages have already been addressed by the CASE tool designers. Moreover, a CASE tool supports the standardization of information systems development within an enterprise.

Still, there are information systems projects that require more flexibility in terms of modeling language syntax, graphical presentations, and semantics of modeling language interactions. The Telos meta modeling language has been developed to address these concerns. Its implementation in ConceptBase [Jarke et al. 1995] , a meta data management system based on the integration of deductive and object−oriented technologies, supports an Internet−based architecture intended to support flexible and goal−oriented distributed cooperation in modeling projects.

A Brief History of Meta Modeling

In the mid−1970s, several semiformal notations supporting the development of information systems were developed. The use of some of these became standard practice in the 1980s, especially entity−relationship diagrams for data modeling and dataflow diagrams for function modeling. More recently, object−oriented methods have added notations for behavior modeling, such as Statecharts, giving a broader picture of the specification and an easier mapping to implementations in languages like C++ or Java.

It was recognized early on that managing large specifications in these notations posed serious problems of inconsistency, incompleteness,maintenance, and reuse. Conceptual modeling languages incorporate ideas from knowledge representation, databases,and programming languages to provide the necessary formal foundation for users with limited mathematics background.

In early 1980s, Sol Greenspan was the first to apply these ideas to requirements engineering, when he formalised SADT notation in the RML language [Greenspan&Mylopoulos 94].This was a precursor to numerous attempts worldwide. Initially, these languages embodied a fixed ontology in which requirements engineering could be described. As early as 1984, it was recognized that modeling formalisms must be customizable.

Jeff Kotteman and Benn Konsynski proposed a basic architecture that included a meta meta model (M2−model for short) as the basis for using different notations within a development environment [Kottemann & Konsynski 84]. ISO's Information Resource Dictionary System (IRDS) [IRDS1990] standard generalized this idea to propose an architecture that combines information systems use and evolution.

The *Instances and scenarios level* within IRDS contains objects which cannot have instances. Examples are data, processes, system states, measurements and so on. Objects may have attributes and they may have classes (residing in the model level). During design, when the information system and therefore the instances do not yet exist, this level also contains scenarios of the intended use of the system.

The *Models level* of IRDS represents the classes of the objects at the instance level. Those classes define the schema (attributes, properties) of the instance level objects as well as rules for manipulating these objects. At the same time the classes are themselves instances of the schema defined at the modeling language level.

At the *Modeling languages level*, meta classes define the structure of the objects (classes) at the model level. In other words, a model is instantiated from the meta classes of the modeling language level. The modeling language level can be used to define specific graphical notations and their interrelationships.

Finally, the *M2−model level* contains meta meta classes (M2−classes). They are classes with instances at the modeling language level. Multiple modeling languages are possible by appropriate instantiations from these M2−classes. Moreover, the dependencies between the multiplelanguages can be represented as attributes between M2−classes in the M2−model level.

Modeling languages do not just have a programming language syntax which needs to be customized. The customization should also address graphical conventions of the modeling formalisms; for example, the mobile phone developer Nokia employs more than 150 method variants in terms of notation, graphics, and ways−of−modeling. Moreover, the correct usage of each formalism and the consistency of models that span across different modeling formalisms should be definable.

Since the late 1980s, more dedicated M2−models have been developed. In parallel, the need to have generalized languages dedicated to meta modeling and method engineering was recognized by

several people. In several iterations, a number of European projects jointly with the group of John Mylopoulos at the University of Toronto developed the language Telos [Mylopoulos & Borgida 90] which generalized RML to provide a meta modeling framework which integrates the three perspectives of structured syntax, graphical presentation, and formal semantics.

However, early attempts to implement the full Telos language (as in the first version of ConceptBase [Jarke & Jeusfeld 89]) showed that its semantics was still too complicated for efficient repository support based on known technologies. Three parallel directions were pursued by different, but interacting and partially overlapping groups of researchers.

The MetaEdit environment developed at the University of Jyväskylä [Kelly & Lyytinen 96] is a good example of an effort focusing on graphics−based method engineering, i.e. the graphical definition of graphical modeling formalisms.

Starting from early experiences with ConceptBase in the DAIDA project [Jarke & Rose 88, Jarke & Mylopoulos 92] the Semantic Index System developed in ESPRIT project ITHACA [Constantopoulos & Jarke 95] focused on ann efficient implementation of the structurally object−oriented aspects of the Telos language. It may be worth noting that the recently announced Microsoft Repository [Bernstein & Harry 97] has generalized such an approach to full object orientation based on Microsoft's Common Object Model.

Three Basic Modeling Methodologies

Modeling processes proceed along three dimensions: representational transformation, domain knowledge acquisition, and stakeholder agreement. Existing methodologies tend to emphasize one of these dimensions over the others: the modeling *notations*, the available *knowledge* within a specific domain, or the *people* involved in the analysis project. All three methodologies have long histories, with little interaction between them. All of them use multiple modeling perspectives but the purpose of these and therefore the integration strategies
are quite different.

Notation−oriented methods manifest their assistance in the set of modeling notations they offer. Their philosophy can be characterized by the slogan "In the language lies the power". Examples of notation−oriented methods are structured analysis approaches, as, e.g., Modern Structured Analysis (MSA) [Yourdon 89], and object−oriented techniques, as, e.g., the Unified Modeling Language (UML) [Fowler & Scott97]. A large number of CASE tools in the market offer graphical editors to develop models of the supported notations and check the balancing rules that must hold between models of different notations. The notations as well as the constraints are hard−coded within the tools and are not easily customizable by users.

A completely different strategy is employed by the *domain−oriented analysis methods*. For a specific application domain, e.g., public administration or furniture industry, they offer a predefined set of reference models. Reference models describe typical data, processes and functions, together with a set of consistency tests which evaluate relationships between the models. Reference models represent the knowledge collected in multiple analysis projects within a particular domain: "In the knowledge lies the power". The reuse of reference models can strongly reduce the analysis effort. However, it can be inflexible since the user can tailor the notations, the constraints or contents only to the degree foreseen by the developers of the reference models, or completely loses the help of the method.

The ARIS Toolset[ARIS 96] offers a platform for working with reference models. It also offers hard−coded constraint checks within and across the models. These tests are programmed individually and new tests can be added manually, without a coherent theory, even though the concept of event−driven process chain (EPC) provides a semi−formal understanding [Scheer 94].

Goal− and team−oriented approaches specifically address the objective to capture requirements from multiple information sources and to make arising conflicts productive.They incorporate stakeholder involvement and prescribe general process steps rather than notations or contents: "In

the people lies the power". Prominent examples include IBM's JAD (Joint Application Design) [August 91] and SSM (Soft Systems Methodology) [Checkland 89]. In these methods highly skilled group facilitators animate the participants, guide the analysis process and keep an eye on the compliance with the specified analysis goals. The general idea is to get as much information as possible from different sources in a short time.

Teamwork remains very informal to enhance creativity. Neither notations nor analysis goals are predefined by the methods but specified by the participants according to the actual problem to be solved. To accommodate the change of goals during project execution, the customization of analysis goals and notations is required even during a running project. Very few supporting tools are available beyond simple groupware tools. The main reason for this dilemma is the high degree of customizability the tools must offer. They must be extensible towards new notations and flexible enough to support changing analysis goals.

The design of a true multi−language, team−oriented and knowledge−intensive modeling system has to address the following goals:

The system should include a feature to define and interrelate specialized conceptual modeling languages in an cost−effective way. The language should reflect the modeler's need of key concepts types and their interpretation of those concepts.

The system should be extensible at any time. When the need for a new concept type occurs, it should be possible to include it into the conceptual modeling language definition in terms of language constructs, graphical presentation, and semantic constraints.

The system should not only check the syntactic correctness within and between models, but also allow to memorize patterns that indicate semantic errors in the models.The memory of those patterns should be extensible and adaptable to the user's growing experience, thus support organizational knowledge creation [Nonaka 94].

Summary and Outlook

Conceptual modeling requires the use of multiple interdependent languages. Selecting the right collection of languages and focusing the analysis of their interactions is a not trivial task. For example, the mobile phone company Nokia claims to employ more than 150 different notations and/or methods in their software
development processes. In such new application domains, standard languages may very well miss the modeling goal by distracting the modelers to details of notation instead to details of the domain to be modeled.

What lies ahead in the next century? On−going computerisation will probably cover more and more aspects of the public and private life. Devices will become more and more aware of their function and will *negotiate* with other devices the best way of performing a certain task. To do so, they may include a conceptual model about themselves. Besides the observed trend to pre−packaged standard software, this computerisation of consumer devices may drastically change the methods in system development. The models will no longer be buried in the rooms of the development team. The models will become part of more and more intelligent devices.

References

[August 91 J.H. August. *Joint Application Design: The Group Session Approach to System Design.*Yourdon Press, Englewood Cliffs, 1991.

[Bernstein & Harry 97] P.A. Bernstein, K. Harry, P. Sanders, D. Shutt, and J. Zander. The Microsoft repository. In *Proc. of the 23rd Intl. Conf. on Very Large Data Bases (VLDB)*, pages 3−12, Athens, Greece, August 1997.

[Borgida & Mylopoulos 93] A. Borgida, J. Mylopoulos, and R. Reiter. ''... and nothing else changes'': The frame problem in procedure specifications. In Proc. of the Fifteenth Intl. Conf. on Software Engineering (ICSE−15)}, May 1993.

[Checkland 89] P.B. Checkland. Soft systems methodology. In J. Rosenhead, editor, *Rational Analysis for a Problematic World*, pages 71−100. John Wiley & Sons, Chichester, 1989.

[Constantopoulos & Jarke 95] P. Constantopoulos, M. Jarke, J. Mylopoulos, and Y. Vassiliou.The software information base: A server for reuse.*VLDB Journal*, 4(1):1−−43, 1995.

[Fowler & Scott97] M. Fowler and K.Scott. *UML Destilled: Applying the Standard Object Modeling Language.* Addison−Wesley, 1997.

[Greenspan & Mylopoulos 94] S. Greenspan, J. Mylopoulos, and A. Borgida. On formal requirements modeling languages: RML revisited. In Proc. of 16th Intl. Conf. on Software Engineering (ICSE), Sorrento, Italy, May 16−21 1994.

[ARIS 96] IDS Prof. Scheer GmbH, Saarbrücken. ARIS−Toolset Manual V3.1, 1996.

[IRDS 1990] ISO/IEC International Standard. *Information Resource Dictionary System (IRDS) − Framework ISO/IEC 10027,* 1990.

[Jarke et al. 95] M. Jarke, R. Gallersdörfer, M.A. Jeusfeld, M. Staudt, and S. Eherer. ConceptBase − a deductive object base for meta data management. *Journal of Intelligent Information Systems*, 4(2):167−192, March 1995.

[Jarke & Jeusfeld 89] M. Jarke, M.A. Jeusfeld. Rule Representation and Management in ConceptBase. *SIGMOD Record*, 18(3):46−51, 1989.

[Jarke & Mylopoulos 92] M. Jarke, J. Mylopoulos, J.W. Schmidt, and Y.Vassiliou. DAIDA: An environment for evolving information systems. *ACM Transactions on Information Systems,* 10(1):1−50, 1992.

[Kottemann & Konsynski 84] J.E. Kottemann and B.R. Konsynski. Dynamic metasystems for information systems development. *Proc. of the 5th Intl. Conf. on Information Systems*, pages 187−204, Tucson, Arizona, November 1984.

[Kelly & Lyytinen 96] S. Kelly, K. Lyytinen, and M. Ross. MetaEdit+: A fully configurable multi−user and multi−tool CASE and CAME environment. In P. Constantopoulos, J. Mylopoulos, and Y. Vassiliou, editors, *Proc. of the 8th Intl. Conf. an Advanced Information Systems Engineering (CAiSE'96)*, pages 1−21, Heraklion, Creta, Griechenland, May 1996. Springer−Verlag, LNCS 1080.

[Mylopoulos & Borgida 90] J. Mylopoulos, A. Borgida, M. Jarke, and M. Koubarakis. Telos: Representing knowledge about information systems. *ACM Transactions on Information Systems*, 8(4):325−362, October 1990.

[Nonaka 94] I. Nonaka. A dynamic theory of organizational knowledge creation. *Organization Science*, (1):14−−37, 1994.

[Scheer 94] A.−W. Scheer. *Business Process Engineering*. Springer−Verlag, 1994.

[Yourdon89 E.Yourdon. *Modern Structured Analysis*. Prentice Hall, Englewood Cliffs, New Jersey, 1989.