A UML-based metamodeling architecture with example frameworks

Marie-Noëlle Terrasse, Marinette Savonnet, George Becker, and Eric Leclercq
Laboratoire LE2I, Université de Bourgogne
B.P. 47870, 21078 Dijon Cedex, France
E-mail: {terrasse,savonnet,becker,leclercq}@khali.u-bourgogne.fr

July 6, 2002

Abstract

Based on a survey of modelers' practice, we propose a UML-based metamodeling architecture in which the two uppermost layers (meta-metamodeling and metamodeling) are organized into a mirroring structure. Using this architecture we can formally define a semantical integration of metamodels. We propose two applications of such metamodel integration: a framework for integrated design and interoperability of information systems, and a framework for the Semantic Web.

1 Introduction

UML modeling is –by definition– a two-step process: first, modelers are expected to define a metamodel (for describing separation of concerns and defining modeling constructs[1]), and second, modelers instantiate their metamodel into a model. Defining a convenient metamodel could be really difficult: it would be necessary to know all earlier metamodel extensions (in order to decide whether an existing metamodel can be used or whether a new one needs to be defined), and it would be necessary to master the manner in which existing metamodels have been built (in order to be able to extend them properly). A first attempt to help modelers is to organize UML-metamodel extensions into an inheritance hierarchy: every extension of a metamodel is a heir of the metamodel being extended.

Actually, modelers do not work directly with the hierarchy of metamodels: information is provided to them in the form of "semi-formal" descriptions (which can be ambiguous but tend to be more readable). These descriptions, which we call modeling paradigms, possibly mix several different languages (e.g., the English language, logics, the set theory, the Z notation, etc.) The modeling process in practice is a three-step process: 1) defining a modeling paradigm (by identifying a modeling paradigm which is close to the needed paradigm, and then building a variant of the existing modeling paradigm), 2) instantiating the modeling paradigm into a metamodel, and 3) instantiating the metamodel into a model. For example, Price's proposal of a UML extension for time models [19] is defined by reference to Kakoudakis's time models [11], and Robbins's proposal for Architecture Description Languages is defined by reference to Medvidovic's proposal [14, 15].

As a consequence, metamodels and models do not provide comprehensive information about the modeling process by which they have been created: all the initial work (in defining modeling paradigms)

[1]A representative example of separation of concerns description can be found in Baumesteir & al.'s UML extension for hypermedia in which extra diagrams are defined for modeling of navigation in a web site. Representative examples of modeling constructs definition can be found in UML extensions for synchronization [9] and Architecture Description Languages [20].

is lost. Thus, some “deep” reasons for the decisions made can be hidden. Our objective is to integrate such information in an OMG-compliant metamodeling architecture.

2 Description of our metamodeling architecture

Our goal is to introduce modeling paradigms into the metamodeling architecture. These modeling paradigms describe –in terms of concepts that are interrelated by constraints– the semantics attached by modelers to the real world. Our hypothesis is that there is a one-to-one correspondence between modeling paradigms and metamodels. This hypothesis leads to the metamodeling architecture described in the following paragraphs and depicted in Figure 1.

A poset of modeling paradigms Modeling paradigms may use a various number of concepts, each of them being described with more or less precision. For example, a modeling paradigm may use a unique concept of *class*, while another modeling paradigm may use different concepts such as *interface*, *abstract class*, and *implementation class*; modeling paradigms may have more or less precise constraints such as *any object must belong to a class* or *any object must belong to one and only one class*. Thus, we define a partial order between modeling paradigms using a subsumption relation: modeling paradigms are informal yet organized in a poset[2] by the partial order between modeling paradigms. We denote by gmp the generic modeling paradigm which is defined by the standard UML semantics: any modeling paradigm is subsumed by gmp.

A mirroring inheritance hierarchy of metamodels Our objective is to build our metamodel layer as a mirror of the poset of modeling paradigms: the generic modeling paradigm gmp is instantiated into the UML metamodel itself (which we denote by mm_{UML}), and all other modeling paradigms are instantiated into specializations of the UML metamodel (by using UML’s extension mechanisms: constraints, tag-values, and stereotypes). Furthermore, we require that each metamodel instantiates a modeling paradigm, each inheritance link between metamodels instantiates a subsumption link between modeling paradigms. Figure 1 presents an example of our mirroring structure in which large grey arrows represent instantiations of modeling paradigms into metamodels.

An example using ADL (Architecture Description Language) Medvidovic & al. [14] describe the C2-style Architecture Description Language in English[3]: “connectors *transmit messages between components, while* components *maintain state, perform operations, and exchange messages with other components via two interfaces (named ”top” and ”bottom”). ... Inter-component messages are either* requests *for a component to perform an operation or* notifications *that a given component has performed an operation or changed state*”. Let call mp_4 the described modeling paradigm. As depicted in Figure 1, the set of elementary concepts, denoted by $\mathcal{E}l^3(mp_4)$, contains concepts *connector*, *component*, *interface*, *message*, etc. The authors also give constraints (which belong to the set of constraints $\mathcal{C}^3(mp_4)$) such as “*components may not directly exchange messages; they may only do so via connectors*”.

Robbins & al. [20] propose an extension of UML for C2-ADL. This extension is an instantiation of the Medvidovic’s modeling paradigm in terms of a metamodel which we denote by mm_4. They define *C2-interface* as a stereotype of the UML *interface* with a tagged value (*top*, *bottom*). *C2-request* and *C2-notification* are defined as stereotypes of the UML *operation* with a constraint forbidding any return

[2]For a definition of a poset see [5]. Such a poset complies with Hehner & al.’s point of view [6]: “*A theory can be presented as a boolean expression. Theories can be ... compared for strength by ordinary implication. Strong theories serve mathematicians who want to prove a lot, but weak theories are better for software engineers who need to implement them*”.

[3]A formal description of C2 in language Z is given in [15].

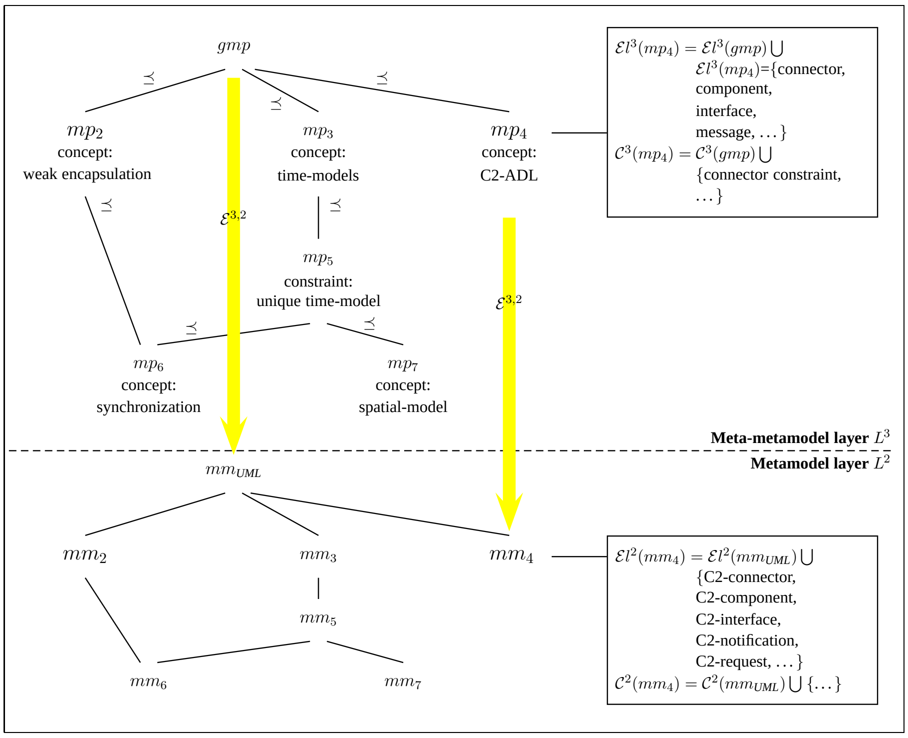
gmp
mp_2 concept: weak encapsulation
mp_3 concept: time-models
mp_4 concept: C2-ADL
mp_5 constraint: unique time-model
mp_6 concept: synchronization
mp_7 concept: spatial-model
$\preceq$
$\mathcal{E}^{3,2}$
$\mathcal{E}l^3(mp_4) = \mathcal{E}l^3(gmp) \bigcup$ $\mathcal{E}l^3(mp_4)$={connector, component, interface, message, …}
$\mathcal{C}^3(mp_4) = \mathcal{C}^3(gmp) \bigcup$ {connector constraint, …}
Meta-metamodel layer L^3
Metamodel layer L^2
mm_{UML}
mm_2
mm_3
mm_4
mm_5
mm_6
mm_7
$\mathcal{E}l^2(mm_4) = \mathcal{E}l^2(mm_{UML}) \bigcup$ {C2-connector, C2-component, C2-interface, C2-notification, C2-request, …}
$\mathcal{C}^2(mm_4) = \mathcal{C}^2(mm_{UML}) \bigcup \{\ldots\}$

Figure 1: Our mirroring structure

value. The tag values *request*, *notification* are used to distinguish requests from notifications. The set of elementary constructs $\mathcal{E}l^2(mm_4)$ is depicted in Figure 1. The set of constraints $\mathcal{C}^2(mm_4)$ contains instantiations of constraints of $\mathcal{C}^3(mp_4)$.

Notation and formalization We denote by L^0, L^1, L^2, and L^3 the instance, model, metamodel, and meta-metamodel layers, respectively. Consistently with this notation, we will attach a superscript (from 0 to 3) to each element that is localized on the corresponding layer of the metamodeling architecture. More details can be found in [21, 22, 23].

A modeling paradigm mp is described by two sets, $\mathcal{E}l^3(mp)$ and $\mathcal{C}^3(mp)$. The set $\mathcal{E}l^3(mp)$ contains descriptions of elementary concepts, while the set $\mathcal{C}^3(mp)$ contains constraints between concepts of $\mathcal{E}l^3(mp)$.

A modeling paradigm mp_1 *is subsumed by* a modeling paradigm mp_2, if both *extended inclusion of concepts* and *subsumption of constraints* are satisfied. Extended inclusion of concepts means that each concept of $\mathcal{E}l^3(mp_2)$ is either a member of $\mathcal{E}l^3(mp_1)$ or a generalization of a concept of $\mathcal{E}l^3(mp_1)$, where a generalized concept may have fewer features than its specialized concept has. Subsumption of constraints means that by using $\mathcal{C}^3(mp_1)$ as a hypothesis, it is possible to prove that each constraint of $\mathcal{C}^3(mp_2)$ holds.

An instantiation function $\mathcal{E}^{3,2}$ is defined in order to build metamodels from modeling paradigms. Let us consider a modeling paradigm mp. $\mathcal{E}^{3,2}$ associates each concept of $\mathcal{E}l^3(mp)$ with one or more elementary components of the UML language. These components are either standard UML constructs or stereotypes which may express constraints of $\mathcal{C}^3(mp)$. We further assume that mp's corresponding metamodel $mm = \mathcal{E}^{3,2}(mp)$ is described by a set of elementary components $\mathcal{E}l^2(mm)$ and a set of constraints $C^2(mm)$. $C^2(mm)$ contains instantiations of some constraints of $\mathcal{C}^3(mp)$, as well as additional constraints due to the instantiation process itself.

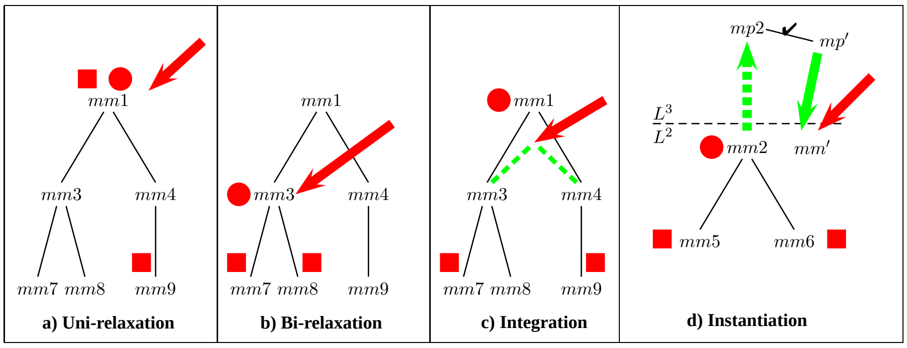

Figure 2: Four cases for building a semantical common ancestor

We say that mp is an *unambiguous* modeling paradigm if $C^2(mm)$, i.e. the instantiated set of constraints of the corresponding metamodel mm, contains no ambiguity.

We say that mp is a *consistent* modeling paradigm if $C^2(mm)$ is consistent.

Two metamodels are said to be *consistent with each other* if the extended union of their sets of constraints contains no contradictions. The extended union of sets of constraints is an union of extended constraints. An extension of constraints is necessary in order to make them meaningful when applied to new concepts.

An *integrated metamodel* is defined by union of sets of concepts and extended union of sets of constraints of two metamodels which are consistent with each other.

3 Using our metamodeling architecture

The increasing interest in UML made it possible to develop a large number of quite different modeling environments, both in the academia and in the industry. These environments provide modelers with numerous tools (for model checking, validation of models against users requirements, refinement of models, generation and testing of executable code, etc.). We believe that by integrating these tools into a metamodeling-based framework it is possible to build more powerful environments. In this section, we first present formal operations on metamodels which rely upon our mirroring structure. Then, we present a framework for integrated design and interoperability of information systems. Finally, we propose a framework for the Semantic Web which is based upon metamodeling and reengineering.

3.1 Formal operations on metamodels

The objective is to define a metamodel which is both unambiguous and a semantically good approximation of the shared part of two given metamodels. By using our metamodeling architecture, we have defined two operations on metamodels: the semantical integration of metamodels and the evaluation of semantical distance between metamodels.

Semantical integration of metamodels The objective is to build a metamodel which can be used as a "semantical common ancestor" for two given metamodels. In our inheritance hierarchy of metamodels, ancestors of a given metamodel are more general than the given metamodel: they do not support that many extensions, and they are not subjected to that many constraints. It is likely that a common ancestor of two given metamodels is a good semantical common ancestor. However, such a common ancestor can be ambiguous. Ambiguities are harmful since they can be lead to inconsistencies when interpreted into different information systems (i.e., into different universes of the discourse). The basis of our strategy is

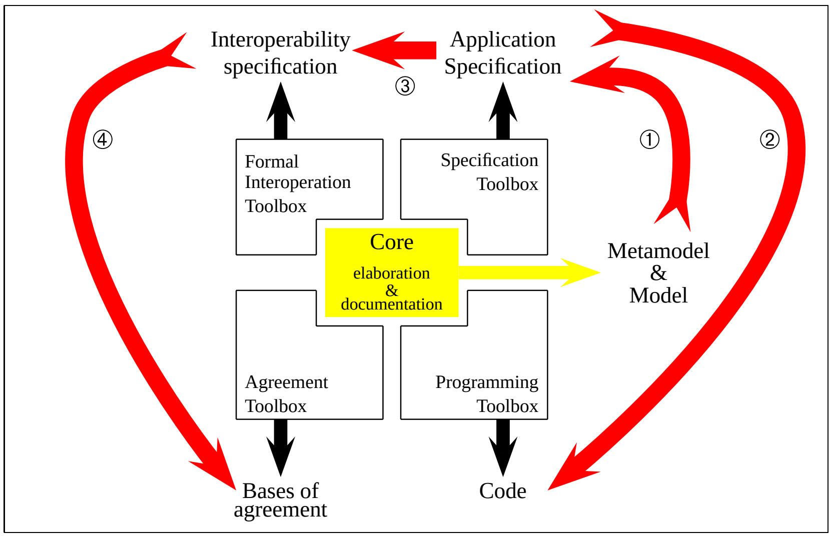
Interoperability specification
Application Specification
③
④
①
②
Formal Interoperation Toolbox
Specification Toolbox
Core
elaboration & documentation
Metamodel & Model
Agreement Toolbox
Programming Toolbox
Bases of agreement
Code

Figure 3: Integrated modeling environment

to use our mirroring structure for building a semantical common ancestor from two given metamodels. Different configurations (i.e., locations of the metamodels to be integrated in the inheritance hierarchy) can be encountered, and this leads to more or less complex solutions, e.g., relaxation (directly choosing a common ancestor in the inheritance hierarchy), integration (of consistent ancestors), instantiation (of a semantically close modeling paradigm).

These different cases are illustrated in Figure 2 in which the given metamodels are depicted by dark-grey squares and the semantical common ancestor is pointed at by a dark-grey thick arrow. In all these cases, the first common ancestor in the inheritance hierarchy is depicted by a dark-grey circle. In part (c) of the figure, the formal integration of metamodels is depicted by dashed light-grey lines. In part (d) of the figure, the instantiation function of a modeling paradigm into a metamodel is depicted by a thick light-grey arrow and its inverse function by a dashed thick light-grey arrow, while a line with a "tick" mark represents subsumption between modeling paradigms.

Measurement of a semantical distance between modeling paradigms Due to our construction process, we can guarantee nominal quality of the semantical common ancestor that we propose, as well as nominal quality of our instantiation function. However, semantic quality of the common ancestor is not guaranteed. In order to be able to evaluate the semantic quality, we need to measure semantic distance between formal metamodels. Our idea for such an evaluation is to build a measure of semantic distance as a weighted sum of elementary distances between corresponding elements of metamodels (i.e., corresponding constructs and corresponding constraints). The main difficulty in such an approach is to determine corresponding pairs of elements. For that, we use our mirroring structure of modeling paradigms and metamodels: our definition of subsumption of modeling paradigms induces a partition of elementary concepts and constraints. Such a partition at the meta-metamodel level induces –by using our instantiation function $\mathcal{E}^{3,2}$– a partition of UML constructs and UML constraints at the metamodel level. See [21] for more details.

In the following sections, we present two frameworks which use our semantical integration of metamodels for interoperability of information systems and the Semantic Web, respectively.

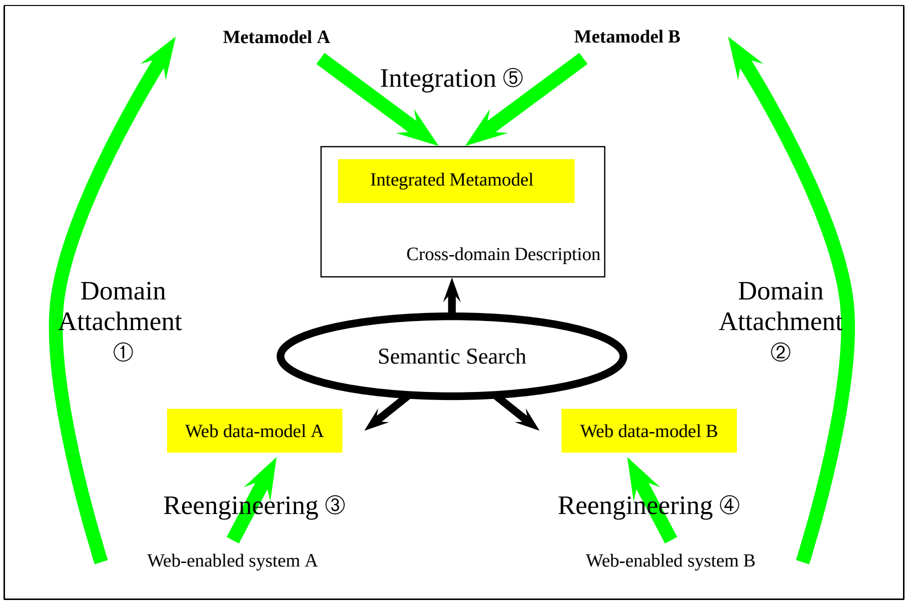

Figure 4: Our Framework for the Semantic Web

3.2 Integrated design and interoperability of information systems

The evolution of major modeling environments seems to inherently proceed towards the integration of formal methods with programming tools. Analogously, building common knowledge (called a basis of agreement) that is to be shared by interoperable systems can be carried out –by formal operations– at the metamodel level and then instantiated at the model and instance levels. Thus we propose to provide modelers with integrated environments for Analysis & Design, as well as for interoperability. Such an integrated environment should be used as described in Figure 3 (in which processes are depicted by numbered dark-grey thick arrows):

A *core component* is used for elaboration of modeling paradigms, metamodels, models and their documentations. A *specification toolbox* encompasses various tools for formal operations on models[4]. By using these tools, modelers can produce a reliable specification of an information system from the previous description (arrow ①). A *programming toolbox* is then used for generation and testing of executable code based upon the specification of an information system (arrow ②, see for example [13, 17]). A *formal interoperation toolbox* produces an abstract basis of agreement –in terms of a metamodel– from a set of specifications of interoperating information systems (arrow ③). Our semantical common ancestor of metamodels (Section 3.1) can be used as an abstract basis of agreement. An *agreement toolbox* is used for generation of actual bases of agreement from abstract bases of agreement (arrow ④): this process is based upon instantiation.

3.3 A metamodeling approach to the Semantic Web

Our metamodeling architecture can be used as a basis for a new approach to the Semantic Web, in which: 1) each domain is described top-down by using a domain library which contains both metamodels and models, 2) web data-model of the involved information systems are produced bottom-up by using systematic reengineering under the control of a domain description, 3) narrowly focused domain descriptions can be constructed separately and then integrated for cross-domain applications.

[4]Formal operations on models can be model checking [7], validation of models against user requirements [2, 10], test generation [1], refinement of models, etc.

The framework we propose encompasses the following processes which are depicted by numbered arrows in Figure 4. The *attachment process* (numbers ①, ②) establishes a dependency between a web-enabled information system and a metamodel which belongs to a library of domain descriptions. The assumption of existence of such a referring metamodel for any web-enabled system may appear rather restrictive. Yet due to the development of UML as a standard tool for high-level system descriptions (see OMG's profiles [18], Cook's prefaces [4], and UML-based ontologies [3]) this assumption is reasonable (or will soon become reasonable). The *reengineering process* (numbers ③, ④) is a reversal of Koch's hypermedia UML-based modeling methods [8, 12]. This reengineering process collects a representative sample of navigation paths and expresses them as a web data-model of the corresponding information system (in terms of well balanced [16] Class and Collaboration Diagrams). The *integration process* (number ⑤) produces an integrated metamodel from which the cross-domain description is built.

4 Conclusion

Our metamodeling strategy has been established on the basis of modelers' behavior and developed in the direction of analysis & design evolution towards more abstract approaches and integration of formal operations. In any case, the modeling process we propose is likely to lead to a new organization of domain modeling: modelers being responsible for "local semantics" (i.e., for describing their own application domain as a variation of an existing domain description), while domain experts being responsible for global semantics (i.e., for validating semantical dependencies between domain descriptions and defining bases for measurement of semantical distances between domain descriptions).

Our short-term work on this subject is to check metamodel integration on different examples from various application domains in order to tune the different parameters of our formal operations. Furthermore, we have to refine the organization of domain descriptions and particularly the architecture of our domain library.

References

[1] P. AMMANN and P.E. BLACK. Abstracting Formal Specifications to Generate Software Tests via Model Checking. In *Proceedings of the 18th Digital Avionics Systems Conference, DASC'99, Saint Louis, Missouri, USA*, 1999. Available at URL www.nist.gov.

[2] K. ANDROUTSOPOULOS. The Reactive System Development Support Tool. Technical report, King's College, Department of Computing, 1999. Available at URL http://www.dcs.kcl.ac.uk/pg/kelly.

[3] K. BACLAWSKI, M. KOKAR, P. KOGUT, L. HART, J. SMITH, W. HOLMES, J. LETKOWSKI, and M. ARONSON. Extending UML to Support Ontology Engineering for the Semantic Web. In *Proceedings of the International Conference on UML, UML'01, Toronto, Canada*, 2001.

[4] S. COOK, A. KLEPPE, R. MITCHELL, B. RUMPE, J. WARMER, and A.C. WILLS. Defining UML Family Members Using Prefaces. In C. Mingins and B. Meyer, editors, *Proceedings of "Technology of Object-Oriented Languages and Systems", TOOLS 32*, pages 102–114. IEEE, November 1999.

[5] G. GRÄTZER. *Lattice Theory, First Concepts and Distributive Lattices*. W.H. Freeman, 1971. ISBN 0-7167-0442-0.

[6] E. HEHNER and I.T. KASSIOS. Theories, Implementations, and Transformations. In *Formal Specification and Development in Z and B*, pages 1–21, 2002. Proceedings of the 2nd International Conference of B and Z Users, BZ'2002, France, Springer Verlag, LNCS 2272, Invited paper.

[7] C. HEITMEYER, J. KIRBY, B. LABAW, and R. BHARADWAJ. SCR: A Toolset for Specifying and Analyzing Software Requirements. In *Proceedings of the 10th Annual Conference on*

Computer-Aided Verification, CAV'98, pages 526–531, Vancouver, Canada, 1998. Available at URL chacs.nrl.navy.mil/SCR.

[8] R. HENNICKER and N. KOCH. A UML-based Methodology for Hypermedia Design. Technical Report 0004, Ludwig-Maximilians University of Munich, Germany, 2000.

[9] J.-L. HERRERO, F. SANCHEZ, F. LUCIO, and M.T. BONILLA. Changing UML Metamodel in Order to Represent Concern Separation. ECOOP'00 Workshop 14 on Defining a Precise Semantics for UML, Sophia Antipolis, France, June 2000.

[10] R.D. JEFFORDS and C. HEITMEYER. An Algorithm for Strengthening State Invariants Generated from Requirements Specifications. In *Proceedings of the 5th International Symposium on Requirements Engineering, RE'01*, Toronto, Canada, August 2001. IEEE.

[11] I. KAKOUDAKIS. *The TAU Temporal Object Model*. Mphil thesis, UMIST, UK, 1996.

[12] N. KOCH and M. WIRSING. Software Engineering for Adaptative Hypermedia Applications. In *Proc. of the 8th Int. Conference on User Modeling, Sonthofen, Germany*, 2001.

[13] A. LEDECZI, M. MAROTI, G. KARSAI, J. GARRETT, C. THOMASON, G. NORDSTROM, J. SPRINKLE, and P. VOLGYESI. The Generic Modeling Environment. In *Proceedings of the WISP'2001 Conference*, 2001.

[14] N. MEDVIDOVIC and D.S. ROSENBLUM. Assessing the Suitability of a Standard Design Method for Modeling Software Architectures. In *Proceedings of the 1st IFIP Working Conference on Software Architecture, San Antonio, Texas, USA*, pages 161–182, 1999.

[15] N. MEDVIDOVIC, R.N. TAYLOR, and Jr. E.J. WHITEHEAD. Formal Modeling of Software Architectures at Multiple Levels of Abstraction. In *Proceedings of the California Software Symposium 1996*, pages 28–40, 1996.

[16] C. NICOLLE and M.N. TERRASSE. RISOM: Towards Comprehensive Reengineering of Information Systems with UML. In *Proceedings of the International Conference on Reengineering Technologies for Information Systems, Retis'01*, 2001.

[17] G. NORDSTROM, J. SZTIPANOVITS, G. KARSAI, and A. LEDECZI. Metamodeling-Rapid Design and Evolution of Domain-Specific Modeling Environments. In *Proceedings of the IEEE Conference and Worshop on Engineering of Computer-Based Systems, ECBS'99*, pages 68–74, 1999. Available at URL computer.org/proceedings/ecbs/0028.

[18] A UML Profile for CORBA, OMG Report 99-08-02, 1999. Available at URL http://www.omg.org, Version 1.0, August 2, 1999.

[19] R. PRICE, K. RAMAMOHANARAO, and B. SRINIVASAN. Spatio-temporal Extensions to Unified Modeling Language. In *Proceedings of the 10th International Worshop on Database and Expert Systems Applications*, pages 460–461. IEEE, September 1999.

[20] J.E. ROBBINS, N. MEDVIDOVIC, D.F. REDMILES, and D.S. ROSENBLUM. Integrating Architecture Description Languages with a Standard Design Method. In *Proc. of the 1998 Int. Conference on Software Engineering*, pages 209–218. IEEE, April 1998.

[21] M.-N. TERRASSE. A Metamodeling Approach to Evolution. In H. Balsters, B. de Bruck, and S. Conrad, editors, *Database Schema Evolution and Meta-Modeling*. Springer-Verlag, LNCS 2065, ISBN 3-540-42272-2, 2001. 9th International Workshop on Foundations of Models and Languages for Data and Objects, Schloss Dagstuhl, Germany, Sep. 2000.

[22] M.-N. TERRASSE and M. SAVONNET. Formalization of the UML Metamodel: An Approach Based Upon the Four-Layer Metamodeling Architecture. ECOOP'00 Workshop 14 on Defining a Precise Semantics for UML, Sophia Antipolis, France, June 2000.

[23] M.-N. TERRASSE, M. SAVONNET, and G. BECKER. A UML-metamodeling Architecture for Interoperability of Information Systems. In *Proceedings of the International Conference on Information Systems Modelling, ISM'01*, 2001. Available at URL http://www.fee.vutbr.cz/UIVT/ism/ISM_programElEdition.htm.