UML-based Frameworks for Engineering of Interoperable Information Systems

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

March 9, 2003

Abstract

We propose a four-layer metamodeling architecture in the context of UML metamodeling. This allows modelers to use a three-step modeling process: 1) giving an informal description of the universe of the discourse (in terms of modeling paradigms); 2) defining a corresponding UML dialect (in terms of metamodels); 3) describing a model of an information system by using the chosen dialect. By using particular properties of our metamodeling architecture, we can define formal and semantical operations on metamodels, e.g. integration of metamodels and measure of semantical distance between them. Such operations are then used as a core of two innovative frameworks. The first framework allows to combine formal and informal approaches for both engineering and interoperability of information systems. The second framework uses metamodeling and reengineering in order to reuse know-how that has been established by researchers in interoperability of information systems in the Semantic Web.

1 Introduction

The role of metamodels in information system engineering has been studied for a long time. Finally, a consensus emerged (e.g., with UML and OMG) in which:

- metamodels form the core of metamodeling architectures by implementing two abstraction mechanisms: abstraction by projection (i.e., separation and combination of concerns [2, 3, 18, 26]), and abstraction by conceptualization (e.g., the Model Driven Architecture approach emphasizes abstraction by conceptualization by using metamodels in a manner analogous to the use of models in the classical modeling).
- metamodels define languages (or dialects [21]) for model descriptions. Recent examples of such languages can be found in UML extensions for synchronization [11] and Architecture Description Languages [25].

- metamodels describe semantics of application domains: metamodels serve as domain ontologies.

UML-based metamodeling architectures use the UML metamodel to support two abstraction mechanisms. First, the UML metamodel is used to define a modeling language which satisfies specific needs of a given application domain. Second, the UML metamodel is used to define diagrams necessary for description of an application belonging to a given domain.

In such metamodeling architectures, information system engineering has been conceived as a two-step process: first, modelers choose a metamodel; and second, modelers instantiate their metamodel into a model. In practice, directly searching for a convenient metamodel could be really difficult. It would be necessary to know all available extensions of a candidate metamodel (in order to decide whether an existing metamodel can be used or whether a new metamodel needs to be defined). It would also be necessary to fully understand the manner in which existing metamodels have been built (in order to be able to extend them properly). One possibility to avoid this difficulty is to reduce metamodel proliferation by providing modelers with standardized UML's metamodel extensions for specific application domains: this is achieved by OMG's profiles [23, 22]. Another possibility is to organize metamodels into a hierarchy [5, 7]: every extension of a metamodel is an heir of the metamodel being extended.

When studying modelers' practice, we notice that modelers do not work directly with metamodel descriptions: information is provided to them in the form of "semi-formal" descriptions (which can be ambiguous but tend to be more readable). We call such a description a *modeling paradigm*. The modeling process –in practice– is a three-step process:

- defining a modeling paradigm (by identifying a modeling paradigm which is reasonably close to the needed paradigm, and then building a variant of such a modeling paradigm),
- instantiating the chosen modeling paradigm into a metamodel,
- instantiating the obtained metamodel into a model.

For example, Price's proposal [24] of a UML extension for time models is defined by reference to Kakoudakis's time models [12], and Robbins's proposal for Architecture Description Languages is defined by reference to Medvidovic & al.'s proposal [15, 16].

Due to the above, metamodels and models do not provide comprehensive information about the actual modeling process: all the initial work (in defining modeling paradigms) is lost. Our objective is to integrate such information into a metamodeling architecture which could then evolve into a comprehensive framework for information system engineering.

In the rest of this paper, we first provide an overview of our metamodeling architecture which encompasses both metamodels and modeling paradigms (Section

2). We then present two example frameworks based upon our metamodeling architecture: a framework for engineering and interoperability of information systems (Section 3), and a framework for the Semantic Web (Section 4). The conclusion (Section 5) discusses our ongoing work.

2 Overview of our metamodeling architecture

In order to help modelers to obtain the maximum benefits from metamodels, we introduce modeling paradigms into our metamodeling architecture. These modeling paradigms describe –in terms of concepts that are interrelated by constraints– the semantics that modelers assign to the real world. Our hypothesis is that there is nearly always, in practice, 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.

2.1 A poset of modeling paradigms

Modeling paradigms are described informally yet they are organized in a poset[1]. Modeling paradigm descriptions possibly mix several different languages: the English language, logics, the set theory, the Z notation, etc. 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*. We define a partial order for modeling paradigms by using a subsumption relation between them.

We denote by gmp the generic modeling paradigm which corresponds to the standard UML semantics. We restrict ourselves to the set of modeling paradigms that are subsumed by gmp: we denote this set by $Restrict_{MP}$. We denote by $\preceq$ the partial order among modeling paradigms.

In Figure 1 we depict some modeling paradigms of $Restrict_{MP}$: mp_7 with a concept of spatiality is subsumed by mp_5. mp_5 itself has a constraint of time model uniqueness and is subsumed by mp_3 supporting several time models. There is no subsumption between mp_3 and mp_4 (mp_4 supports the C2 Architecture Description Language).

2.2 A mirroring inheritance hierarchy of metamodels

Our objective is to build the metamodel layer of our architecture as a mirror of the poset of modeling paradigms: the generic modeling paradigm gmp is instantiated

[1]We use Gratzer's definition of posets [8]: posets (also called partially ordered sets) are "*sets equipped with partial ordering relations*".

into the UML metamodel itself (which we denote by mm_{UML}), and all other modeling paradigms are then 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 some modeling paradigm, and that each inheritance link between metamodels instantiates a subsumption link between modeling paradigms.

For example, in Figure 1 the general modeling paradigm gmp is instantiated into the UML metamodel mm_{UML} while mp_4 is instantiated into a metamodel mm_4. Large grey arrows represent instantiations of modeling paradigms into metamodels.

2.3 An example using ADL (Architecture Description Language)

Medvidovic & al. [15] describe the C2-style Architecture Description Language in English[2]: "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 us call by mp_4 the modeling paradigm described above. As depicted in Figure 1, the description of mp_4 includes concepts (e.g., *connector, component, interface, message*) and constraints (e.g., "*components may not directly exchange messages; they may only do so via connectors*").

Robbins & al. [25] propose an extension of UML's metamodel for C2-ADL. Their extension is an instantiation of the Medvidovic's modeling paradigm in terms of a metamodel which we denote by mm_4. They define $\ll C2 - interface \gg$ as a stereotype of the UML interface with a tagged value (*top, bottom*). $\ll C2 - operation \gg$ (which can be either a request for operation or a notification message) is defined as a stereotype of the UML operation with a constraint forbidding any return value. The tagged values *request, notification* are used to distinguish requests from notifications. The set of elementary UML constructs of mm_4 is depicted in Figure 1. The set of UML constraints of mm_4 contains instantiations of constraints of modeling paradigm mp_4.

2.4 Semantical integration of metamodels

Our objective is to define a metamodel which can achieve a good semantical approximation of concepts shared between two given metamodels. The basis of our strategy is to use our mirroring structure for building a convenient common semantical ancestor from the two given metamodels, i.e., a common ancestor which is semantically close to these metamodels. Different configurations (i.e., relative positions of two metamodels to be integrated in the inheritance hierarchy) can be

[2] A formal description of C2 in language Z is given in [16].

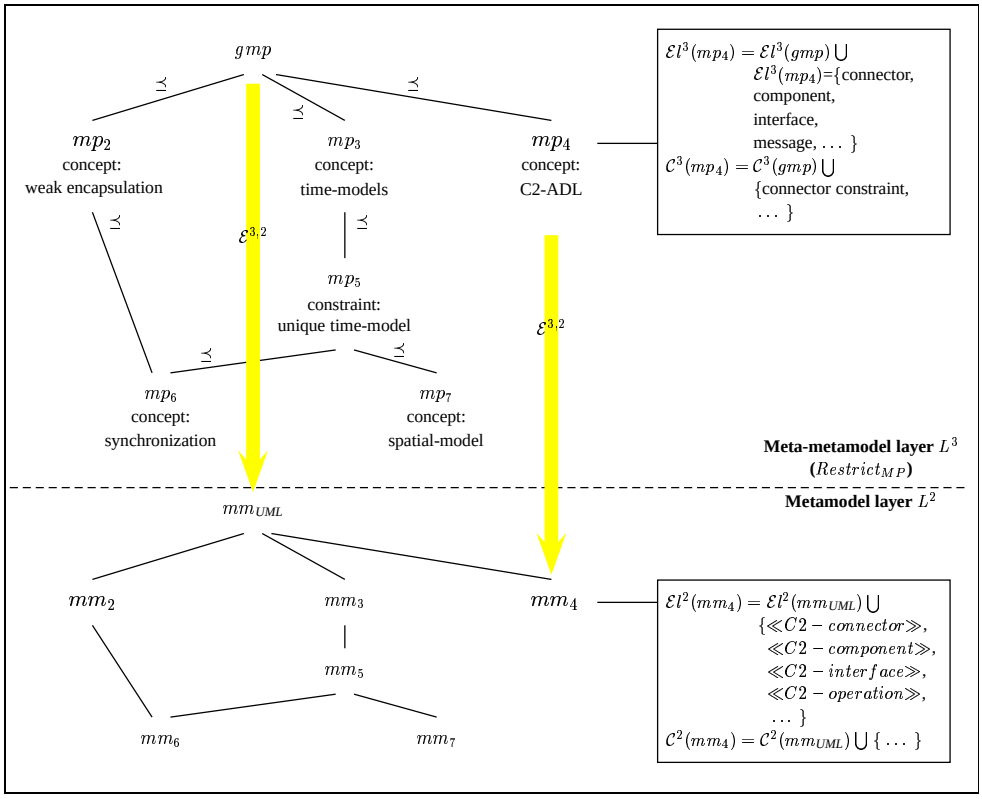

Figure 1: Our mirroring structure of meta-metamodel and metamodel layers

encountered. This leads to more or less complex solutions: *relaxation* (directly choosing a common ancestor in the inheritance hierarchy), *formal integration* (of consistent ancestors), *instantiation* (of a semantically close modeling paradigm chosen from the poset of modeling paradigms). See [28] for details. Due to our construction process, we can guarantee nominal quality of the proposed semantically integrated metamodel, as well as nominal quality of our instantiation function. However, semantical quality of such an integrated metamodel is not yet guaranteed. We believe that semantical quality of semantically integrated metamodels depends on the quality of our mirroring structure (and, ultimately, on the quality of the poset of modeling paradigms). We propose to delegate the responsibilities of ensuring the semantical quality of the poset of modeling paradigms to domain experts. In addition to the experts' participation in the overall quality of the poset of modeling paradigms, we intend to measure semantical distance between metamodels in order to evaluate quality of semantically integrated metamodels.

2.5 Semantical distance between modeling paradigms

In order to be able to evaluate semantical quality, we need to measure semantical distances between metamodels. We defi ne 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 principal question in such an approach is how to identify corresponding pairs of elements. In the rest of this section, we outline a generic method for determining pairs of such elements. Elementary distances between corresponding elements, as well as weights used for combining these elementary distances, are not discussed in the general case: they need to be fine-tuned in the context of a specific application domain. We propose to delegate the responsibilities of defining these elementary distances and weights to domain experts. For a formal presentation of our method please refer to [27].

Let us consider a simplified version of Robbins & al.'s metamodel (metamodel mm_4 in Figure 1). Metamodel mm_4 is a specialization of mm_{UML}, and mp_4 is subsumed by gmp. For evaluation of semantical distance between mm_4 and mm_{UML} we introduce the following sets:

- At the meta-metamodel level (mp_4 and gmp), the set of common concepts:

$$Com(mp_4, gmp) = \mathcal{E}l^3(gmp)$$

- At the meta-metamodel level (mp_4 and gmp), the set of specialized concepts:

$$Spe(mp_4, gmp) = \{interface, message\}$$

- At the meta-metamodel level (mp_4 and gmp), the set of new concepts:

$$New(mp_4, gmp) = \{connector, component\}$$

- At the metamodel level (mm_4 and mm_{UML}), the pairs of initial UML constructs and specialized constructs:

$$\begin{aligned} Spe_{UML}(mm_4, mm_{UML}) = \{&(\ll C2 - operation \gg, operation),\\ &(\ll C2 - interface \gg, interface),\\ &(\ll C2 - component \gg, class),\\ &(\ll C2 - connector \gg, class)\}. \end{aligned}$$

  We note that two new concepts of mp_4 are instantiated as specializations of the UML class: their instantiations belong to $Spe_{UML}(mm_4, mm_{UML})$. Thus, $New_{UML}(mm_4, mm_{UML}) = \emptyset$.

An example evaluation of the mismatch between UML constructs and the above stereotypes is given in the following paragraph (see Figure 2). For such an evaluation, we use a measure of elementary distances with only three possible values (*low*, *middle* and *high*) from which a more precise measure can be derived.

- Specialized constructs belonging to Spe_{UML} introduce:

Stereotype	UML construct	Extension mechanism	Spe_{UML} low	Spe_{UML} middle	Spe_{UML} high	Add_{UML} low	Add_{UML} middle	Add_{UML} high
C2-operation	operation	tagged value and constraint	•	•				
C2-interface	interface	tagged value and constraint	•			•		
C2-component	class	constraint	••		••			
C2-connector	class	constraint	•					

Figure 2: The first step in measurement of mismatch between the UML metamodel and Robbins & al.'s metamodel

- A stereotype $\ll C2 - operation \gg$ of UML operations with a tagged value (*request*, *notification*) and a constraint $\ll C2 - operation \gg$ *has no return value*. The tagged value is evaluated as *low* and the constraint is evaluated as *middle*.
- A stereotype $\ll C2 - interface \gg$ of UML interfaces with a tagged value (*top*, *bottom*) and a constraint $\ll C2 - interface \gg$ *operations must have stereotype* $\ll C2 - operation \gg$. Both the tagged value and the constraint are evaluated as *low*. The constraint does not belong to Spe_{UML} (since it is a constraint between two stereotypes) but rather to the set of additional UML constraints Add_{UML}.
- A stereotype $\ll C2 - component \gg$ of class with constraints[3]:
  - $\ll C2 - component \gg$ *do not contain features*: evaluated as *high*,
  - $\ll C2 - component \gg$ *implements exactly two interfaces (with stereotype* $\ll C2 - interface \gg$*) top and bottom*: evaluated as *middle* and belonging to Add_{UML},
  - *requests travel upward only* and *notifications travel downward only*: both evaluated as *low*,
  - *each* $\ll C2 - component \gg$ *has exactly one instance in the running system*: evaluated as *high*.
- A stereotype $\ll C2 - connector \gg$ of class with a constraint: $\ll C2 - connectors \gg$ *are limited to binary associations*: evaluated as *low*.

3 A Framework for Engineering and Interoperability of Information Systems

The increasing interest in UML made it possible to develop a large number of various modeling environments, both in the academia and in the industry. These environments provide modelers with numerous tools (for design, validation, and implementation of information systems). The evolution of many Analysis & Design environments seems to inherently proceed towards the integration of formal methods with programming tools. Analogously, building common knowledge (called

[3]In order to make our example sufficiently short, we do not use all the constraints given in [25].

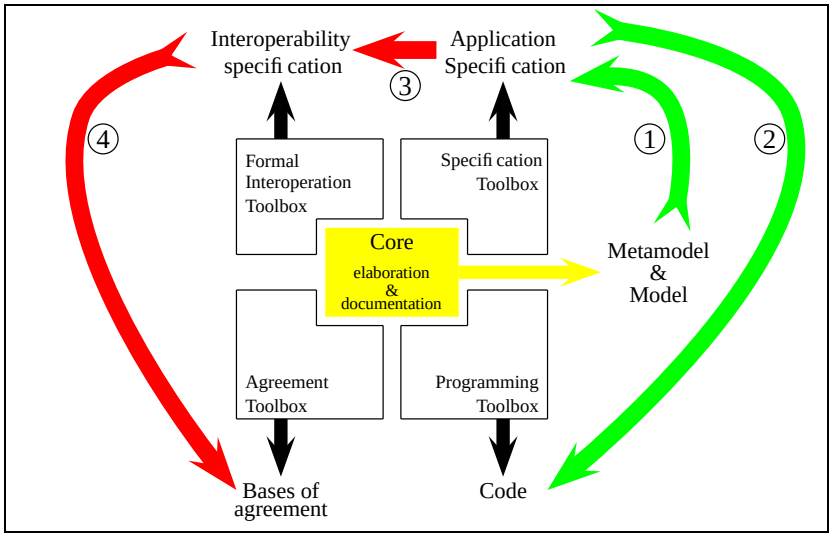

Figure 3: Integrated environment for Analysis & Design and interoperability

a basis of agreement) that is shared by interoperable systems can be carried out –by formal operations– at the metamodel level and then instantiated at the model and instance levels. We therefore seek to provide modelers with integrated environments for engineering of interoperable information systems. Such an integrated environment can be used as outlined in Figure 3 (where processes are depicted by numbered dark-grey thick arrows). A *core component* is used for elaboration of modeling paradigms, metamodels, models and their documentation. A *specification toolbox* encompasses various tools for formal operations on models. By using such tools, modelers can produce a reliable specifi cation of an information system from the description constructed in the core component (Arrow 1). A *programming toolbox* is then used for generation and testing of executable code based upon the specifi cation of an information system (Arrow 2). A *formal interoperation toolbox* produces an abstract basis of agreement –in terms of a metamodel– from a set of specifi cations of interoperating information systems (Arrow 3). An *agreement toolbox* is used for generation of actual bases of agreement from abstract bases of agreement (Arrow 4). In the following paragraphs, we describe these components in more details (except for the formal interoperation toolbox which is discussed in Section 2).

The core component is dedicated to elaboration of metamodels, models and their documentation. This core component is supported by our mirroring structure. It allows modelers to choose a convenient initial modeling paradigm or to defi ne their own modeling paradigm when necessary, to instantiate it into their metamodel, and fi nally to create their own model. For example, Clark & al.'s [4] have proposed an environment for metamodel construction. Such an environment includes formal tools for model validation.

The programming toolbox is used for generation and testing of executable code. Most of the commercial products for UML-based modeling include toolboxes that provide modelers with tools for generation of code (e.g., C++ or Java) from a model of the system. In many cases, code generation is carried out from an unvalidated model: an error in the model is thus propagated into the code. By producing code from a complete and consistent specification, and by using tests produced by our specification toolbox, it is possible to guarantee the quality of the resulting code.

For example, Norstrom & al. [20] have proposed a UML-based environment, called Model Integrated Computing (MIC), for generation of programs. MIC encompasses a partial validation of the model. They have further extended MIC into a Generic Modeling Environment [14].

The formal interoperation toolbox produces an abstract basis of agreement from a set of specifications of interoperating information systems. Such an abstract basis of agreement is expressed in terms of metamodels. The formal interoperation toolbox is based upon our semantical integration of metamodels: abstract bases of agreement are produced by semantical integration of the metamodels of information systems involved. For a given group of information systems, several abstract bases of agreement can be proposed (depending on the strategy used for building instantiated metamodels). By using our semantical distance between metamodels, we can help users to evaluate the proposed bases of agreement and to choose the most suitable one.

The role of domain experts is particularly relevant for this toolbox: they have to validate the part of the poset of modeling paradigms that describes their application domain and they have to determine the needed parameters (weights and elementary distances) for the semantical distance measure.

The agreement toolbox is used for generation of actual bases of agreement from abstract bases of agreement. We use abstract bases of agreement in the same way Gruber [9] uses ontologies of representation (i.e., ontologies which capture representation primitives but do not define their content). Defining the content of an actual basis of agreement is a process similar to the instantiation of a metamodel into a model. The main difference is that the information to be modeled is –in general– much more difficult to identify.

4 A Framework for the Semantic Web

With the worldwide development of web-enabled information systems, the need for efficient data access integrators (such as portals) and comprehensive interoperating services is urgent. The idea of a Semantic Web, as "*a web talking to machines*" has been widely accepted. According to Euzenat [6], manipulating only unstructured information has become an obsolete option: information models are needed (e.g., RDF, UML, XML) in order to balance "*formalized knowledge*" against "*actual*

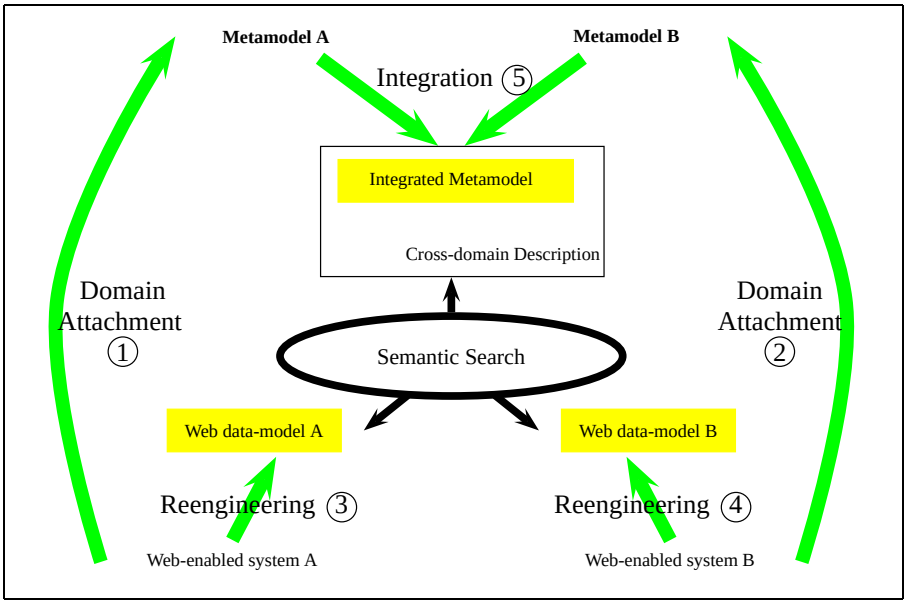

Figure 4: Our Framework for the Semantic Web

informal contents". In any case, even by using the pre-existing information models, it appears difficult to build the Semantic Web from scratch without reusing the know-how that has been established and refined for interoperability [17].

We propose a framework for developpement of the Semantic Web. Since web sites used in searches are not necessarily described at a conceptual level (and even when they are described at this level, it may be done so using different models), we propose to systematically build structural descriptions of web-enabled systems by using reengineering techniques. The main advantage of such a systematic reengineering is the possibility to provide homogeneous descriptions of web-enabled information systems: any web-system can be described using the same information model (i.e. UML) and the same description strategy [29]. 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.

The framework we propose encompasses the following processes which are depicted by numbered arrows in Figure 4:

- The *attachment process* (arrows 1, 2) establishes a dependence between a web-enabled information system and a metamodel which belongs to a library of domain descriptions. Requiring 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[4] this requirement is reasonable (or will soon become reasonable). Domain libraries also cover the model and instance levels of descriptions. They are organized as UML packages. Each package is a narrowly targeted description of a specific domain. These packages can be mutually linked in order to produce cross-domain descriptions. The aim of our on-going work is to precisely define the attachment component of our framework.

- The *reengineering process* (arrows 3, 4) is a reversal of Koch's hypermedia UML-based modeling methods [10, 13]. 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 [19] Class and Collaboration Diagrams):

  - First, we navigate the web site in order to extract a graph of inter-object links. Such a graph is used as a representative sample of the whole navigation space. We propose this graph as a Collaboration Diagram for the model to be built.
  - Second, by using our domain integrator, we produce a "minimal" Class Diagram for the model being built.
  - Third, we measure the adequacy of the Collaboration Diagram in combination with the Class Diagram. As long as its adequacy is insufficient, we keep modifying the Class Diagram and repeating the above steps.

- The *integration process* (arrow 5) produces an integrated metamodel from which the cross-domain description can be built. The integration component of our framework has been presented in Section 2.

5 Conclusion

Our metamodeling architecture takes advantage of knowledge of modelers' behaviors, abstract approaches to information system engineering, and formal methods. Modeler's behavior is represented by a poset structure of modeling paradigms at the meta-metamodel level. Abstraction is implemented by metamodels which form the core of our architecture. Cohesion of the two uppermost layers of our architecture is guaranteed by our mirroring structure of the poset of modeling paradigms and the inheritance hierarchy of metamodels. Therefore, we may "induce" properties of modeling paradigms from properties that have been formally defined on metamodels (e.g., ambiguity, consistency).

[4] See, for example, OMG's profiles [23], Cook's prefaces [5], and UML-based ontologies [1].

We further define formal operations on metamodels. We also extend these "purely" formal operations into semantical ones: semantical integration of metamodels and a measure of semantical distance between metamodels. These semantical operations form the basis of two frameworks which apply metamodeling to information system interoperability and the Semantic Web. We continue our work on several technical questions related to this project:

- Providing modelers with a convenient method for using our mirroring structure in the form of a library of domain descriptions (i.e., for describing modeling paradigms and instantiating corresponding metamodels).
- Providing domain experts with a method for adaptating our generic measure of semantical distance between metamodels to the experts' application domains.
- Defining generic rules for quality measurement of the sub-poset of modeling paradigms corresponding to closely related application domains.
- Defining what is the required functionality of agreement toolboxes (for generation of actual bases of agreement from abstract bases of agreement) in our framework for integrated engineering and interoperability of information systems.
- Defining an attachment process of our framework for the Semantic Web. The main anticipated problem is how to determine limitations and prerequirements of such a process.

Finally, we believe that the extensive use of our proposal would open a "political issue", namely the need for a new organization of domain modeling. Modelers would be responsible for "local semantics" (i.e., for describing their own application domain as a variation of an existing domain description). Domain experts would be responsible for the global semantics (i.e., for validating semantical dependencies between domain descriptions).

References

[1] 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.

[2] Jean Bezivin. On Different Interoperability Modes in Software Engineering: the Case of Modeling Activities at OMG. In *Proceedings of Software Engineering'98*, Paris, France, December 1998.

[3] Ruth Breu, Radu Grosu, Franz Huber, Bernhard Rumpe, and Wolfgang Schwerin. Systems, Views and Models of UML. In Martin Schader and Axel Korthaus, editors, *The Unified Modeling Language, Technical Aspects and Applications*, pages 93–109. Physica Verlag, 1998. Available at URL http://www.cs.york.ac.uk/puml.

[4] T. Clark, A. Evans, S. Kent, and P. Sammut. The MMF Approach to Engineering Object-Oriented Design Languages. In *Proceedings of the Workshop on language, descriptions, Tools and Applications, LDTA01*, 2001.

[5] Steve Cook, Anneke Kleppe, Richard Mitchell, Bernhard Rumpe, Jos Warmer, and Alan Cameron 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.

[6] Jérôme Euzenat (Scientific Coordinator). Research Challenges and Perspectives of the Semantic Web. In *Report of the EU-NFS Strategic Workshop, Sophia-Antipolis, France*, 2001.

[7] E. D. Falkenberg and J.L. Han Oei. Meta Model Hierarchies from an Object-Role Modeling Perspective. In *Proceedings of the 1^{st} International Conference on Object-Role Modeling, ORM-1, Magnetic Island, Australia*, 1994.

[8] George Grätzer. *Lattice Theory, First Concepts and Distributive Lattices*. W.H. Freeman, 1971. ISBN 0-7167-0442-0.

[9] T. Gruber. A Translation Approach to Portable Ontology Specification. *Knowledge Acquisition*, 5:199–220, 1993.

[10] Rolf Hennicker and Nora Koch. A UML-based Methodology for Hypermedia Design. Technical Report 0004, Ludwig-Maximilians University of Munich, Germany, 2000.

[11] José Luis Herrero, Fernando Sanchez, Fabiola Lucio, and Miguel Toro 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.

[12] I. Kakoudakis. *The TAU Temporal Object Model*. Mphil thesis, UMIST, Manchester, UK, 1996.

[13] Nora Koch and Martin Wirsing. Software Engineering for Adaptative Hypermedia Applications. In *Proceedings of the 8^{th} International Conference on User Modeling, Sonthofen, Germany*, 2001.

[14] Akos Ledeczi, Miklos Maroti, Gabor Karsai, Jason Garrett, Charles Thomason, Greg Nordstrom, Jonathan Sprinkle, and Peter Volgyesi. The Generic Modeling Environment. In *Proceedings of the WISP'2001 Conference*, 2001.

[15] Nemad Medvidovic and David 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.

[16] Nemad Medvidovic, Richard N. Taylor, and Jr. E. James Whitehead. Formal Modeling of Software Architectures at Multiple Levels of Abstraction. In *Proceedings of the California Software Symposium 1996*, pages 28–40, 1996.

[17] Sergey Melnik and Stefan Decker. A Layered Approach to Information Modeling and Interoperability on the Web. In *Proceedings of the Workshop on the Semantic Web, ECDL'2000*, 2000.

[18] Hafedh Mili, François Pachet, Ilham Benyahia, and Fred Eddy. Workshop Summary. In *Proceedings of OOPSLA'95 Workshop on Metamodeling in OO*, pages 105–110, Austin, TX, USA, October 1995.

[19] 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.

[20] Greg Nordstrom, Janos Sztipanovits, Gabor Karsai, and Akos Ledeczi. Metamodeling-Rapid Design and Evolution of Domain-Specifi c 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.

[21] Ileana Ober. Diffi culties in Defi ning Precise Semantics for UML. ECOOP'00 Workshop 14 on Defi ning a Precise Semantics for UML, Sophia Antipolis, France, June 2000.

[22] Roadmap for the Business Object Initiative: Supporting Enterprise Distributed Computing, OMG Report 98-10-09. Available at URL http://www.omg.org.

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

[24] Rosanne Price, Kotagiri Ramamohanarao, and Bala Srinivasan. Spatiotemporal Extensions to Unifi ed Modeling Language. In *Proceedings of the 10th International Worshop on Database and Expert Systems Applications*, pages 460–461. IEEE, September 1999.

[25] Jason E. Robbins, Nemad Medvidovic, David F. Redmiles, and David S. Rosenblum. Integrating Architecture Description Languages with a Standard Design Method. In *Proceedings of the 1998 International Conference on Software Engineering*, pages 209–218. IEEE, April 1998.

[26] Bernhard Schätz and Franz Huber. Integrating Formal Description Techniques. In Jeannette M. Wing, Jim Woodcock, and Jim Davies, editors, *FM'99 – Formal Methods*, volume 2 of *LNCS 1709*, pages 1206–1225. Springer Verlag, September 1999. Available at URL http://www4.informatik.tu-muenchen.de/papers.

[27] Marie-Noëlle 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. 9^{th} International Workshop on Foundations of Models and Languages for Data and Objects, Schloss Dagstuhl, Germany, September 2000.

[28] MN Terrasse, M. Savonnet, G. Becker, and E. Leclercq. A UML-Based Metamodeling Architecture with Example Frameworks. WISME 2002, Workshop on Software Model Engineering, Dresden, Germany, Available at URL http://www.metamodel.com/wisme-2002/terrasse.pdf, 2002.

[29] F. Vauglin. Statistical Representation of Relative Positional Uncertainty for Geographical Linear Features. In *Proceedings Workshop CASSINI*, Paris, France, April 1997.