UML-based Metamodeling for Information System Engineering and Evolution

Marie-Noëlle Terrasse, Marinette Savonnet,
George Becker, and Eric Leclercq

Laboratory LE2I - University of Burgundy - France
E-mail: marie-noelle.terrasse@u-bourgogne.fr,
marinette.savonnet@u-bourgogne.fr,
george.becker@khali.u-bourgogne.fr,
eric.leclercq@u-bourgogne.fr

Abstract. In modelers' practice metamodels have become the core of UML-based metamodeling environments: metamodels form the basis of application domain descriptions, and they are instantiated into models. In the context of information system engineering and interoperability, we have developped two operations on metamodels: metamodel integration and measure of semantical distance between metamodels. In this paper, we explore application of these operations to information systems' evolution.

1 Introduction

Information system engineering is becoming an increasingly complex process which has to take into account intricacies of information systems, deployment environments, and users' demands [17]. In order to cope with such demanding requirements, information system engineering has been using sophisticated abstraction mechanisms. Metamodel-based abstraction (e.g., OMG's four layer metamodeling architecture and UML-based metamodeling environments) uses a metamodel to describe an application domain; a model is then instantiated from a metamodel. In this section, we offer a dual application of metamodeling: first for information system engineering, and second for information system evolution.

1.1 Metamodeling and Information Systems

Metamodeling environments that have been proposed, either in industry or in academia, provide modelers with a core component for first defining a metamodel[1] (i.e., for application domain description), then defining a model as an instance of this metamodel, etc. In order to avoid metamodel proliferation it is necessary to "reuse" application domain descriptions: Lester & al. call it *domain reuse* [12], the OMG has defined specific *profiles* for various application domains [18, 19]. In practice, the modeling process encompasses three steps:

[1] E.g., Clark & al.'s proposal [3].

1. Describing informally an application domain: such a description can use various notations and languages. We call such a description a *modeling paradigm.*
2. Instantiating a modeling paradigm into a metamodel.
3. Instantiating the obtained metamodel into a model.

As a consequence of such a modeling process, the metamodel and model levels do not provide sufficient information about the modeling process. All of the initial work, i.e., searching for an appropriate modeling paradigm, has been lost. We present an example (on time models) of a lost modeling paradigm: Project TAU [26], of the Timelab laboratory, introduced a time model defined in the English language. Such a definition uses mathematical properties for the time axis[2], defines the calendar to be used[3], etc. Kakoudakis & al. [9, 25] used the TAU time model as a basis for a temporal extension of UML. They called their extension TUML, which is a part of TAU Project. The authors proposed several stereotypes such as ≪date≫, ≪time≫, ≪timestamp≫, ≪timepoint≫, ≪period≫. Price & al. [22] later used Kakoudakis' proposal in order to build a spatio-temporal extension of UML, called STUML. They proposed –in terms of stereotypes– various time models which are based on the TAU proposal. Each temporal stereotype encompasses a *specification box*[4] that defines a time model, a time unit, an interpolation hypothesis, etc. Yet, TAU time models –which are the actual basis of these UML-extensions– have not been described in UML and thus, do not appear in a classical UML-based metamodeling architecture.

In the next section, we argue that modeling paradigms have also a role to play in the context of information system evolution.

1.2 Metamodeling and Model Evolution

Evolution of information systems has for a long time presented a significant challenge and has necessitated the introduction of model mapping. Generally, model mapping has to be carried out under demanding conditions such as inclusion of business and application-domain changes, integration of development and deployment environment features, etc. Several authors propose to control model evolution by using schema invariants which are enforced by rules [4, 5, 10, 20, 21]. Beyond a certain point, a model cannot evolve under constraints of these invariants: it is a case of *paradigm shift*[5] [24]. As depicted in Figure 1, it may be necessary to use metamodel reengineering (in order to take into account –at the

[2] TAU project: "*The time axis is considered to be discrete, linear, totally ordered and bounded at both ends.*"

[3] TAU project: "*The model adopts the Gregorian calendar and supports its default granularities.*"

[4] As an example, they describe a specification box for a temporal attribute of population density: time is one-dimensional, its interpolation hypothesis requires that values do not change from one measurement to another, time is linear and regular, and time measurement frequency is annual.

[5] Note that such a paradigm shift occurs at the model level.

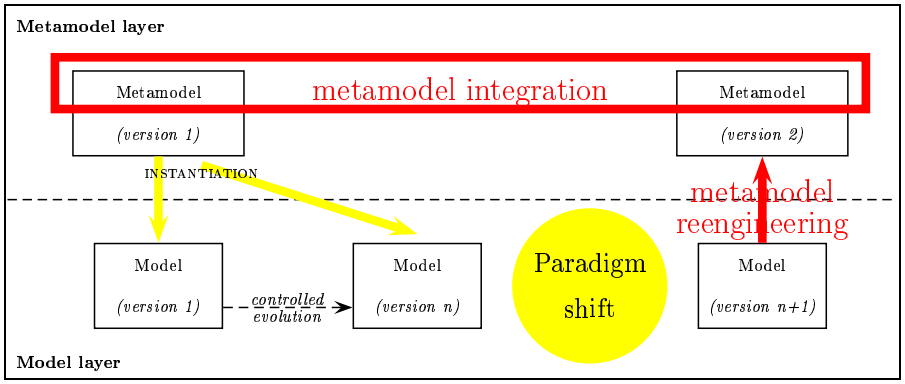

Fig. 1. Paradigm shift as a discontinuity in model evolution

metamodel level– changes that violate invariants) and metamodel integration (in order to provide a sound semantical basis to information system operation[6]).

Another perspective, which is emerging with the Model Driven Approach, views paradigm shift as a business (or domain) input which makes the modeling paradigm to evolve. Nurcan & al. [16] present an example of an electrical provider which has to evolve from a position of a monopoly to being exposed to pressures of competition. As a consequence, its business process needs to change in order to integrate new business goals (e.g., profitability). As depicted in Figure 2, it is necessary to integrate metamodels which implement source and target modeling paradigms. Several abstract descriptions of paradigm shift have been proposed in terms of graph rewriting (Sprinkle & al. [24]), meta-templates (De Miguel & al. [15]), logical constraints (Caplat & al. [2]). Such abstract descriptions of the paradigm shift are presented at the metamodel level. Etien & al. [5] also propose a metamodel level description in terms of two metamodels: a generic (abstract) metamodel and a specific metamodel.

We propose a metamodeling architecture that encompasses modeling paradigms together with metamodels and models. Section 2 presents our metamodeling architecture. Section 3 introduces operations on metamodels and discusses their importance for information system evolution. Finally, Section 4 outlines our metamodeling perspective on information system evolution.

2 Our metamodeling architecture

We introduce modeling paradigms into our metamodeling architecture in order to have a comprehensive trace of the modeling process. Such 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 exists –in

[6] We have discussed the need of a sound semantical basis in [29].

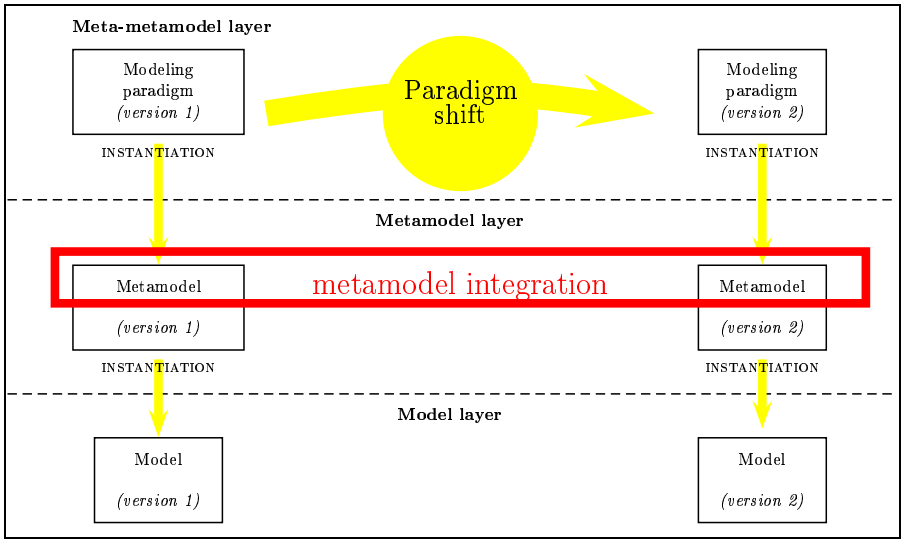

Fig. 2. Paradigm shift at the meta-metamodel level

practice– a one-to-one correspondence between modeling paradigms and metamodels. This hypothesis leads to a mirroring structure which is described in the following paragraphs and depicted in Figure 3.

2.1 A poset of modeling paradigms

Modeling paradigms are described informally; such descriptions may mix several different languages. Modeling paradigms are described in terms of concepts. Each modeling paradigm may use different 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 multiple concepts of class, such as *interface*, *abstract class*, and *implementation class*. Constraints may be defined among concepts of a modeling paradigm. 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 between modeling paradigms by using a subsumption relation[7], which we denote by $\preceq$. We thus obtain a poset[8] of modeling paradigms

[7] A modeling paradigm mp_1 is subsumed by a modeling paradigm mp_2 ($mp_1 \preceq mp_2$), if both extended inclusion of concepts and subsumption of constraints apply. Extended inclusion of concepts means that each concept of mp_2 is either a concept of mp_1 or a generalization of a concept of mp_1, where a generalized concept may have fewer features than its specialized concept has. Subsumption of constraints means that using the set of constraints of mp_1 as a hypothesis, it is possible to prove that each constraint of mp_2 holds.

[8] We use Gratzer's definition of posets [7]: posets (i.e., partially ordered sets) are "*sets equipped with partial ordering relations*".

which agrees with Hehner & al.'s point of view [8]: "*A theory can be presented as a boolean expression. Theories can be ... compared for strength by ordinary implication.*".

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 $Restrict_{MP}$. For each modeling paradigm *mp*, we denote by $\mathcal{E}l^3(mp)$ and $\mathcal{C}^3(mp)$ the sets of concepts and constraints of *mp*, respectively.

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 into the UML metamodel itself (which we denote by mm_{UML}), and all other modeling paradigms of $Restrict_{MP}$ are instantiated into specializations of the UML metamodel (by using UML's extension mechanisms: constraints, tag-values, and stereotypes). For each modeling paradigm, new concepts and constraints are instantiated into UML-constructs and OCL-constraints. Generally, a new concept is instantiated by stereotyping an existing UML-construct. In some cases an existing concept can be specialized instead. New constraints are instantiated into OCL-constraints or (for some of them) introduced as additional constraints into an existing construct.

For each metamodel *mm*, we denote by $\mathcal{E}l^2(mm)$ and $\mathcal{C}^2(mm)$ the sets of UML-constructs and OCL-constraints of *mm*, respectively.

In order to guarantee a perfect mirroring of a poset of modeling paradigms, we require that each metamodel instantiates a modeling paradigm, and that each inheritance link between metamodels instantiates a subsumption link between modeling paradigms.

As depicted in Figure 3, the general modeling paradigm *gmp* is instantiated into the UML metamodel mm_{UML}. Instantiations of modeling paradigms into metamodels are represented by large arrows. Modeling paradigms and metamodels shown in Figure 3 are described in more details in Section 2.3 and Section 2.4.

2.3 Examples from hypermedia application domains

Hypermedia application domains describe web sites in terms of their contents and their presentation. We have chosen three different examples:

Koch & al.'s [11] (UWE project) propose a modeling paradigm for hypermedia which we denote by mp_2. It describes main features of a web site, e.g., *navigational space* and *navigational structure*. The navigational space describes which classes are offered for visit, together with associations (links available for navigation) between these classes. A navigational structure describes which type of navigation is offered by each link (guided tour, index, etc.).

Koch & al.'s modeling paradigm for hypermedia, mp_2, is instantiated into a metamodel mm_2. Its navigational space is described by a diagram encompassing

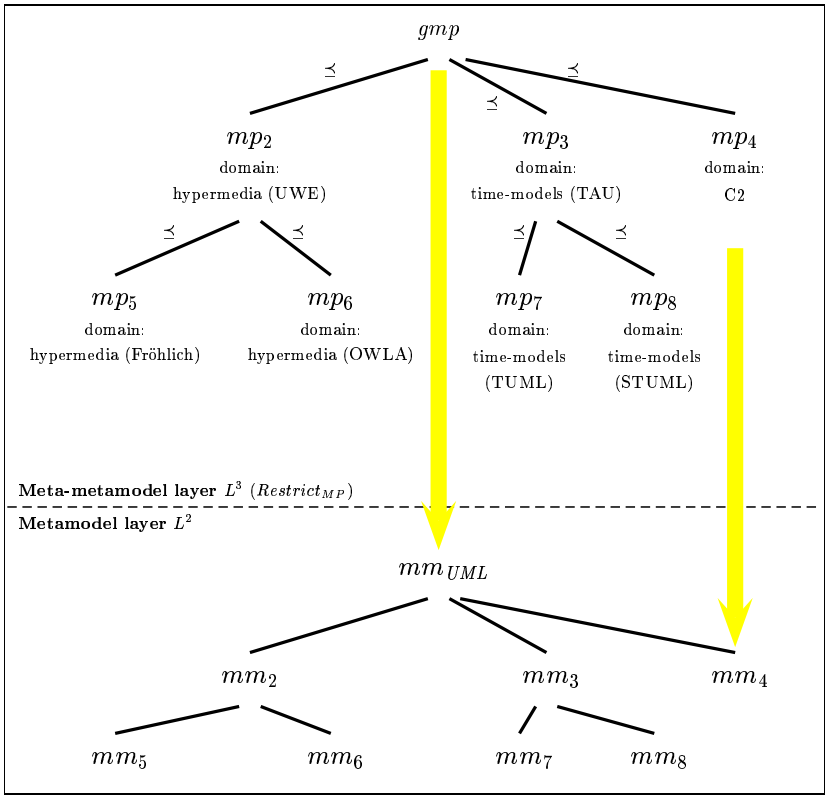

Fig. 3. Our mirroring structure of meta-metamodel and metamodel layers

stereotypes $\ll NavigationalClass\gg$, $\ll PresentationalClass\gg$, $\ll GuidedTour\gg$, among others.

Fröhlich & al. [6] propose another modeling paradigm for hypermedia which we denote by mp_5. It is similar to mp_2 except that additional restrictions are imposed on mp_5: a restriction to binary associations, and a limitation of types of navigation (in the navigational structure) depending on multiplicities of associations (in the navigational space).

Fröhlich & al.'s modeling paradigm for hypermedia, mp_5, is instantiated into a metamodel called mm_5. An OCL-constraint allows only binary associations.

Alatalo & al. [1] propose a modeling paradigm for mobile-aware hypermedia (OWLA project) which we denote by mp_6. mp_6 uses the concept of *conditional association* in order to discriminate between associations with guarded navigability (depending on users's location) and unguarded associations.

Alatalo & al.'s modeling paradigm for mobile-aware hypermedia, mp_6, is instantiated into a metamodel mm_6. A stereotyped association is also introduced; we denote it by $\ll CondAssociation\gg$.

Koch & al.'s modeling paradigm mp_2	
set of concepts $\mathcal{E}l^3(mp_2)$	$\mathcal{E}l^3(mp_2) = \mathcal{E}l^3(gmp) \bigcup \{NavigationalSpace,\ NavigationalStructure,$ $StaticPresentationalSpace,\ \ldots\}$
set of constraints $\mathcal{C}^3(mp_2)$	$\mathcal{C}^3(mp_2) = \mathcal{C}^3(gmp) \bigcup \{Consistency\ of\ ClassDiagram\ with$ $NavigationalSpace \ldots\}$
Koch & al.'s metamodel mm_2	
set of UML-constructs $\mathcal{E}l^2(mm_2)$	$\mathcal{E}l^2(mm_2) = \mathcal{E}l^2(mm_{UML}) \bigcup \{\ll NavigationalClass\gg,\ \ll GuidedTour\gg,\ \ldots\}$
set of OCL-constraints $\mathcal{C}^2(mm_2)$	$\mathcal{C}^2(mm_2) = \mathcal{C}^2(mm_{UML}) \bigcup \{OCL:\ rules\ for\ using\ the\ above\ stereotypes\}$

Fig. 4. Modeling paradigms and metamodels for hypermedia: Koch & al.

Fröhlich & al.'s modeling paradigm mp_5	
set of concepts $\mathcal{E}l^3(mp_5)$	$\mathcal{E}l^3(mp_5) = \mathcal{E}l^3(mp_2)$
set of constraints $\mathcal{C}^3(mp_5)$	$\mathcal{C}^3(mp_5) = \mathcal{C}^3(mp_2) \bigcup \{binary\ association,\ inter-diagram\ constraint,\ \ldots\}$
Fröhlich & al.'s metamodel mm_5	
set of UML-constructs $\mathcal{E}l^2(mm_5)$	$\mathcal{E}l^2(mm_5) = \mathcal{E}l^2(mm_2)$
set of OCL-constraints $\mathcal{C}^2(mm_5)$	$\mathcal{C}^2(mm_5) = \mathcal{C}^2(mm_2) \bigcup \{OCL:\ binary\ association,$ $OCL:\ inter-diagram\ constraint, \ldots\}$

Fig. 5. Modeling paradigms and metamodels for hypermedia: Fröhlich & al.

2.4 An example of a software component application domain

Medvidovic & al. [13] describe a modeling paradigm, which we denote by mp_4, for an Architecture Description Language called C2. C2-like software architectures consist in software components that exchange messages (called *requests* and *notifications*) by using interfaces (called *top* and *bottom*) which are connected by connectors. Medvidovic & al. describe the C2-style language in English[9]: "connectors *transmit messages between components, while* components *maintain state, perform operations, and exchange messages with other components*

[9] A formal description of C2 in the Z notation is given in [14].

Alatalo & al.'s modeling paradigm mp_6	
set of concepts $\mathcal{E}l^3(mp_6)$	$\mathcal{E}l^3(mp_6) = \mathcal{E}l^3(mp_2) \bigcup \{conditional\ association\}$
set of constraints $\mathcal{C}^3(mp_6)$	$\mathcal{C}^3(mp_6) = \mathcal{C}^3(mp_2) \bigcup \{rules\ for\ using\ conditional\ association\}$
Alatalo & al.'s metamodel mm_6	
set of UML-constructs $\mathcal{E}l^2(mm_6)$	$\mathcal{E}l^2(mm_6) = \mathcal{E}l^2(mp_2) \bigcup \{\ll CondAssociation\gg\}$
set of OCL-constraints $\mathcal{C}^2(mm_6)$	$\mathcal{C}^2(mm_6) = \mathcal{C}^2(mp_2) \bigcup \{OCL:\ rules\ for\ using \ll CondAssociation\gg\}$

Fig. 6. Modeling paradigms and metamodels for hypermedia: Alatalo & al.

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*". As depicted in Figure 3, 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. [23] propose an extension of UML's metamodel for C2. Their extension is an instantiation of Medvidovic's modeling paradigm in terms of a metamodel denoted 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 an 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 3. The set of OCL-constraints of mm_4 contains instantiations of constraints of the modeling paradigm mp_4.

Medvidovic & al.'s modeling paradigm mp_4	
set of concepts $\mathcal{E}l^3(mp_4)$	$\mathcal{E}l^3(mp_4) = \mathcal{E}l^3(gmp) \bigcup \{connector,\ component,\ interface,\ message,\ \ldots\}$
set of constraints $\mathcal{C}^3(mp_4)$	$\mathcal{C}^3(mp_4) = \mathcal{C}^3(gmp) \bigcup \{connector\ constraint,\ \ldots\}$
Medvidovic & al.'s metamodel mm_4	
set of UML-constructs $\mathcal{E}l^2(mm_4)$	$\mathcal{E}l^2(mm_4) = \mathcal{E}l^2(mm_{UML}) \bigcup \{\ll C2-connector\gg,\ \ll C2-interface\gg,$ $\ll C2-operation\gg,\ \ll C2-component\gg,\ \ldots\}$
set of OCL-constraints $\mathcal{C}^2(mm_4)$	$\mathcal{C}^2(mm_4) = \mathcal{C}^2(mm_{UML}) \bigcup \{\ldots\}$

Fig. 7. Modeling paradigms and metamodels for C2 Architecture Description Language

3 Metamodels and information system evolution

Continuity of information system evolution can be guaranted by invariants and guarding rules. Discontinuities occur in information system evolution when invariants are violated. The resulting paradigm shift is a model-level event. As discussed in Section 1, a model-level paradigm shift necessitates metamodel reengineering, i.e., construction of a metamodel from a given model of the information system. If the resulting metamodel is not an extension of the initial metamodel of the information system, we need to integrate these two metamodels in order to provide a sound semantical basis for the information system operation. Thus, metamodel integration can be used whenever information system evolution is carried out "non-continuously".

An extreme case of discontinuity in information system evolution occurs when its application domain changes extensively. In such a case, the modeling para-

digm of the information system evolves. For this reason, we need to build a new metamodel which is no longer an extension of the initial metamodel of the information system. Metamodel integration is then necessary in order to accomplish information system evolution at the modeling paradigm level.

Measure of distance between metamodels can be used to evaluate disimilarity between two metamodels in order to decide whether a metamodel integration is required or not. Furthermore, it would be useful to tune our measure so that each elementary distance relevant to an evolution invariant has a significant effect. As a consequence, our measure could also be used for paradigm shift detection.

In the following sub-sections we present two operations which we have developped on metamodels by using our mirroring architecture: " *metamodel integration* and a measure of *semantical distance* between metamodels.

3.1 Metamodel integration

We need to construct an integrated metamodel; this can be achived by using specific properties of our metamodeling architecture[10]. The central idea of such integration is to build an integrated metamodel as an union of constructs and an union of constraints (whenever no inconsistencies exist). For example, integration of metamodels mm_5 and mm_6 produces a metamodel, denoted by mm_9, which we define as follows:

- $\mathcal{E}\mathbf{l^2(mm_9)} = \mathcal{E}l^2(mm_2) \bigcup \{\ll CondAssociation\gg\}$
- $\mathcal{C}\mathbf{^2(mm_9)} = \mathcal{C}^2(mm_2) \bigcup \{\ OCL: rules\ for\ using\ conditional\ association,$
  $OCL: restriction\ to\ binary\ association,$
  $OCL: inter-diagram\ constraint\ \}$

When it is not possible to build an union of constraints (because inconsistencies occur), we use our mirroring structure in order to produce a suitable integrated metamodel. An integrated metamodel can thus differ significantly from both initial metamodels. In order to evaluate suitability of an integrated metamodel, we have defined a measure of a "distance" between metamodels.

3.2 Measure of distance between metamodels

We build a measure of semantical distance as a weighted sum of elementary distances between corresponding elements of metamodels (i.e., between corresponding constructs and corresponding constraints). The main difficulty in such an approach is how to determine corresponding pairs of elements. For that, we use our partial order among modeling paradigms which enables us to determine common concepts, specialized concepts, and new concepts (common constraints, specialized constraints, and new constraints, respectively).

Let us present an example on metamodels mm_4 and mm_{UML} (see Figure 3). The metamodel mm_4 is a specialization of mm_{UML}. For evaluation of the semantical distance between mm_4 and mm_{UML}, we use corresponding modeling

[10] See [28, 29] for more details.

paradigms, i.e., mp_4 and gmp. We build:
1) the set of concepts common to gmp and mp_4, i.e., $\mathcal{E}l^3(gmp)$;
2) the set of gmp concepts specialized in mp_4, i.e., $\{interface, message\}$;
3) the set of concepts introduced in mp_4, i.e., $\{connector, component\}$. From these three sets, we determine pairs of UML-constructs (for which we need to define elementary distances):

$(\ll C2 - operation \gg, message)$,
$(\ll C2 - interface \gg, interface)$,
$(\ll C2 - component \gg, class)$,
$(\ll C2 - connector \gg, class)$.

In the above example, we can distinguish various types of "elementary distances". First, one of the elements of a pair can be built as a slight variation of the corresponding element, e.g., adding a tagged value (*request,notification*) to the UML-construct operation when defining the $\ll C2 - operation \gg$ stereotype. Second, one of the elements of a pair can be built by adding a weak constraint to the corresponding element. Such an additionnal constraint cannot weaken a fundamental feature of the corresponding element: e.g., a constraint "$\ll C2 - operation \gg$ *has no return value*" added to the UML-construct *operation* to define a $\ll C2 - operation \gg$ stereotype. Third, one of the elements of a pair can be built by modifying a fundamental feature of the corresponding element. For example, the constraint "*each* $\ll C2 - component \gg$ *has exactly one instance in the running system*" induces a major change when going from the UML-construct *class* to the stereotype $\ll C2 - component \gg$.

Elementary distances between corresponding elements, as well as weights assigned to them, are not discussed in the general case: they need to be fine-tuned in the context of a specific application domain. We generally leave to domain experts the responsibilities of defining elementary distances and weights. See [27] for more details.

4 Conclusion

We provide examples –in spatio-temporal information systems– of different levels of information system evolution: instance level, model level, and metamodel level. The instance-level evolution is evolution limited to unit or reference changes (e.g., from hour to year, from Christian to Hijra era, from kilometer to ligth-year). The model-level evolution reflects changes in the underlying topology (e.g., from date to interval for time, from Clark to equatorial coordinates for space). The metamodel-level evolution expresses change of the underlying model of space or time (e.g., from earth to galactic space).

In the context of metamodeling environments, we propose to extend the requirement of continuous evolution to the metamodel level. Both model and metamodel of an information system have to preserve invariants throughout their evolution. The main advantage of such double continuity is a reduction of the paradigm shift impact: minimal metamodel reegineering is required. Evolution at the modeling paradigm may be necessary in case of a significant change in the

business process. In such a case, we propose to determine whether metamodel integration is necessary or not (by using our measure of a semantical distance between metamodels).

References

1. T. Alatalo and J. Peraho. Designing Mobile-aware Adaptive Hypermedia. In *Proc. of the Third Workshop on Adaptive Hypertext and Hypermedia*, 2001.
2. G. Caplat and JL. Sourrouille. Model Mapping in MDA. In *Proc. of the WISME Workshop in Software Model Engineering*, 2002. Germany.
3. T. Clark, A. Evans, S. Kent, and P. Sammut. The MMF Approach to Engineering Object-Oriented Design Languages. In *Proc. of the Workshop on Language, Descriptions, Tools and Applications, LDTA01*, 2001.
4. G. Engels, R. Heckel, J.M. Küster, and L. Groenewegen. Consistency-Preversing Model Evolution through Transformations. In *Proc. of the 5^{th} Int. Conf. on the Unified Modeling Language*. Springer, LNCS 2460, 2002. Germany.
5. A. Etien and C. Salinesi. Towards a Sytematic Definition of Requirements for Software Evolution: A Case-Study Driven Investigation. In *8^{th} CAiSE Int. Workshop on Evaluation of Modeling Methods in Systems Analysis and Design, EMMSAD'03*, pages 65–73, 2003. Austria.
6. P. Fröhlich, N. Henze, and W. Nejdl. Meta-Modeling for Hypermedia Design. In *Proc. of the Second IEEE Metadata Conference, MD97*, 1997.
7. G. Grätzer. *Lattice Theory, First Concepts and Distributive Lattices.* W.H. Freeman, 1971. ISBN 0-7167-0442-0.
8. E. Hehner and I.T. Kassios. Theories, Implementations, and Transformations. In *Formal Specification and Development in Z and B*, pages 1–21, 2002. Proc. of the 2^{nd} Int. Conf. of B and Z Users, BZ'2002, France, Springer Verlag, LNCS 2272, Invited paper.
9. I. Kakoudakis. *The TAU Temporal Object Model.* Mphil thesis, UMIST, Manchester, UK, 1996.
10. W. Kim, N. Ballou, H-T. Chou, J.F. Garza, and D. Woelk. Features of the Orion Objet Oriented Database. In W. Kim and F.H. Lochovsky, editors, *Object Oriented Concepts, Databases, and Applications.* ACM Press, 1989.
11. N. Koch, H. Baumeister, R. Hennicker, and L. Mandel. Extending UML for Modeling Navigation and Presentation in Web Applications. In *Proc. of the Workshop Modeling Web Applications in the UML, UML'00*, 2000.
12. N.G. Lester, F.G. Wilkie, and D.W. Bustard. Applying UML Extensions to Facilitate Software Reuse. In *The Unified Modeling Language - UML'98: Beyond the Notation*, pages 393–405. Springer, LNCS 1618, 1998.
13. N. Medvidovic and D.S. Rosenblum. Assessing the Suitability of a Standard Design Method for Modeling Software Architectures. In *Proc. of the 1^{st} IFIP Working Conf. on Software Architecture, USA*, pages 161–182, 1999.
14. N. Medvidovic, R.N. Taylor, and Jr. E.J. Whitehead. Formal Modeling of Software Architectures at Multiple Levels of Abstraction. In *Proc. of the California Software Symposium 1996*, pages 28–40, 1996.
15. M.A. De Miguel, D. Exertier, and S. Salicki. Specification of Model Transformations Based on Meta Templates. In *Proc. of the WISME Workshop in Software Model Engineering*, 2002. Germany.

16. S. Nurcan, J. Barrios, and C. Rolland. Une méthode pour la définition de l'impact organisationnel du changement. In *Proc. du Congrès informatique des organisations et systèmes d'information et de décision*, 2002.
17. UML4MDA, Response to the omg RFP Infrastructure for UML2.0, Report 2003-01-13. Available at URL http://www.omg.org, January 2003.
18. Roadmap for the Business Object Initiative: Supporting Enterprise Distributed Computing, OMG Report 98-10-09. Available at URL http://www.omg.org.
19. A UML Profile for CORBA, OMG Report 99-08-02, 1999. Available at URL http://www.omg.org, Version 1.0, August 2, 1999.
20. R.J. Peters and T. Özsu. An Axiomatic Model of Dynamic Evolution in Objectbase Systems. *ACM Transactions on Database Systems*, 22 (1):75–114, 1997.
21. C. Pons, G. Baum, and R.D.K. Paige. Model Evolution ans System Evolution. *Journal of Computer Science and Technology Special Issue on Computer Science Researsch: a State of Art*, 2000. Available at URL journal.info.unlpedu.ar/journal.
22. R. Price, B. Srinivasan, and K. Ramamohanarao. Extending the Unified Modeling Language to Support Spatiotemporal Applications. In C. Mingins and B. Meyer, editors, *Proc. of TOOLS 32, Conf. on Technology of Object-Oriented Languages and Systems*, pages 163–174. IEEE, 1999.
23. 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. Conf. on Software Engineering*, pages 209–218. IEEE, 1998.
24. J. Sprinkle and G. Karsai. Defining a Basis for Metamodel Driven Model Migration. In *Proc. of the 9^{th} Int. Conf. and Workshop on the Engineering of Computer-Based Systems*. IEEE, 2002. Sweden.
25. M. Svinterikou and B. Theodoulidis. The Temporal Unified Modelling Language TUML. Technical Report TR-97-1, TimeLab, Department of Computation, U-MIST, UK, 1997.
26. Tau project. Technical report, Timelab, UMIST, UK, 1997. URL http://www.co.umist.ac.uk/ timelab/projects/tau.html.
27. MN. 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} Int. Workshop on Foundations of Models and Languages for Data and Objects, Germany, 2000.
28. MN. Terrasse, M. Savonnet, and G. Becker. An UML-metamodeling Architecture for Interoperability of Information Systems. In *Proc. of the Int. Conf. on Information Systems Modelling, ISM'01*, 2001.
29. MN Terrasse, M. Savonnet, G. Becker, and E. Leclercq. A UML-Based Meta-modeling Architecture with Example Frameworks. WISME 2002, Workshop on Software Model Engineering, Germany, Available at URL http://www.metamodel.com/wisme-2002/terrasse.pdf, 2002.