CREWS Report 98-11

Appeared in
Proceedings of the Eight annual Workshop on Information Technologies and Systems WITS'98. Computer Science and Information Systems Reports TR-19, University of Jyväskylä, Finnland.

CASE Environment Adaptability: Bridging the Islands of Automation

by

K. LYYTINEN*, P. MARTIIN*, J.-P. TOLVANEN*,
M. JARKE*, K. POHL*, K. WEIDENHAUPT*

(*): Department of Computer Science
and Information Systems
University of Jyväskylä, Finland

(**): Lehrstuhl Informatik V, RWTH Aachen
Ahornstraße 55, 52056 Aachen, Germany
+49 (0)241 80 21 501
{haumer,pohl}@informatik.rwth-aachen.de

CASE Environment Adaptability: Bridging the Islands of Automation

Kalle Lyytinen
Pentti Marttiin
Juha-Pekka Tolvanen

Department of Computer Science
and Information Systems
University of Jyväskylä, Finland

Matthias Jarke
Klaus Pohl
Klaus Weidenhaupt

Informatik V
RWTH Aachen, Germany

Abstract

In current CASE environments a user has to choose between efficient computerized support using a fixed methodical framework which may not fit his situation, or the freedom to do what seems appropriate in the given circumstance, but at the cost of losing efficient technological support. In this paper we examine adaptable metamodel based environments as a means to resolve this dilemma. Metamodel based environments provide means to represent and modify knowledge about development products, processes, and representation schemes to improve the designer-task fit. Metamodel based adaptability is not a new innovation. Yet, earlier metamodel based approaches have tended to create islands of automation that focus on improving adaptability either in ontologies, notations, or process definitions. The paper applies a framework which integrates these aspects and thereby increases environment adaptability that covers a wider spectrum of development situations. The metalevel based integration is demonstrated by describing how two metamodel based tools focusing on different aspects of adaptability can be integrated. The resulting integrated environment encompasses both product (ontology/notation) and process aspects.

1 Introduction

Information systems development (ISD) is a knowledge intensive and complex undertaking, and adequate tool support for its improvement has been hard to come by. In fact, the state of the art in support functionality is far from adequate (cf. [10]). In current CASE environments a user has to choose between efficient computerized support using a fixed methodical framework which may not fit his situation, or the freedom to do what seems appropriate in the given circumstance, but at the cost of losing efficient technological support. In this paper we argue that one main reason for this has been a focus on “islands of automation” in thinking of support functionality, which neglects sufficient levels of adaptability in combining tool support. Only through achieving high levels of adaptability that are easy to implement and broad enough in terms of functionality we can expect advances in CASE environments.

The need for adaptability can be explained by differences across development processes, deliverables and varying contingencies at different stages. Therefore ISD methods combined with tools have to be modified to take into account development contingencies such as uncertainty, size, time, resources, and available skills [4], [13], [15], [6]. The need for adaptability is further emphasized because the contingencies are likely to evolve over time within and across projects. Several surveys [20], [8] and case studies [1], [14] confirm the need for higher levels of adaptability. Method developers (e.g. [3]) building standardized methods have also noticed the need for situational adaptability. From these observations two requirements in developing tool environments arise. First, the choice of methods in terms of concepts, notations and processes should be as little as possible constrained by chosen technologies. Second, the environment must be sufficiently flexible in that enough degrees of freedom in tool use are available at any process stage i.e. any possible route to adequate system solution should be supported.

Two questions we will ask in this paper are: 1) how such adaptability can be achieved, and 2) what remains to be done to achieve such adaptability? To the first question we suggest a metamodeling approach, i.e. an approach in which knowledge about the system and its development processes can be changed in the tool environment at one level higher — through metamodels. These metamodels span over method's ontologies, notations, and processes to achieve a higher level in adaptability. This is but one approach in achieving adaptability: the others include control integration through standardized tool interfaces [23] or bus architectures [21]. However, these are insufficient because they neglect the explicit modeling of systems development knowledge.

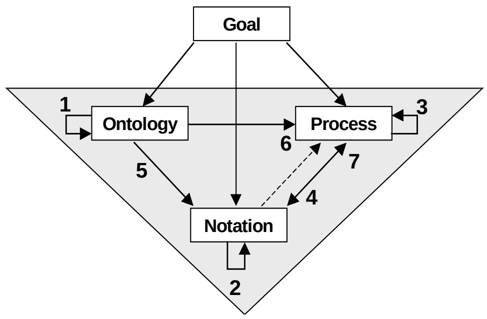
Although metamodeling in itself is not a new innovation, we want to pinpoint that current metamodeling approaches result in "islands of automation" in that they focus primarily on one aspect of the metamodeling area. To overcome this limitation we apply a (meta) framework which shows how these islands can be bridged and how this bridging helps in achieving adaptability of CASE environments. This will answer the second question. The environment adaptability is demonstrated in more detail by describing integration of two metamodel based tools which as "islands of automation" focus on different aspects of adaptability: One in adaptation of ontologies and notations (i.e. product) and the other in adaptation of processes. The environment integration is described at two levels. At the metalevel we discuss on alternatives to integrate product and process based metamodels. At instance level we use the proposed integration schema to demonstrate integration of ISD methods. The resulting integrated environment encompasses both product and process aspects.

The paper is organized as follows. Sect. 2 describes a framework for adaptable metamodel based environments. Sect. 3 describes examples of metamodel based adaptability from both the product and the process side. In Sect. 4 we discuss the alternatives to achieve adaptable integration between different tools and demonstrate integration using the framework. Sect. 5 concludes the paper by suggesting directions for future research.

2 Framework for Metamodel Based Adaptation

We will analyze metamodel based adaptation by applying a framework called diamond model [10]. The model (cf. Fig. 1) consists of two parts: (1) a triangle, which describes a possible space of choices when adapting methods. This space is organized in terms of ontologies, notations and processes; and (2) the goals of an ISD project at the top which drive the choices made in the adaptation triangle. In this paper we focus on the choices inside a method, i.e. triangle part of the framework.

The idea of the triangle shape is to illustrate that these three aspects are neither exclusive, nor orthogonal, i.e. each viewpoint complements the other viewpoints and all are required to yield "complete" adaptability in systems development and its computer support.



Each aspect of the triangle has been modeled and represented independently using various alternative representation schemes and mechanisms (cf. [10] for a survey). These approaches focus on different aspects of method adaptability (arrow 1 to 3 in the Fig. 1) and result in different types of metamodels. This explains why "islands of automation" in terms of computer support have been built. In the following, we first describe each node of the model individually (arrow 1-3 in Fig. 1) and then focus on the relationships between them (arrows 4-7).

2.1 Method Adaptation within a Single Node

2.1.1 Ontological Adaptability

Because it is impossible to represent a system in its full detail methods focus on a small number of abstractions based on a set of generic concepts. The enumeration of the concepts used in the conceptualization is called ontology [24]. An example of such an ontology would be use case based requirements engineering [3] in which concepts like *actor* and *use case* are defined.

Accordingly, ontological adaptation deals with establishing applicable abstractions based on a set of generic concepts. Due to differences in systems, development contingencies and different philosophical positions there exist a wide variety of ontologies, and their concepts. Arguments in favor of different ontological constructs in entity-relationship (ER) modeling are good examples for the needs to different ontologies. An example of ontology adaptation at the detailed level would be adding of an is*A*-relationship between entities. An example of ontological adaptation at higher granularity level would be defining an ontology for use case based requirements engineering.

2.1.2 Notational Adaptability

Notations are used to represent and manipulate ontological constructs. Notations can take various presentation styles including graphical, matrix, tabular, textual, symbolic, or mixed syntaxes. The simplest form of notational adaptability is to offer a selection of a symbol from a set of graphical symbols, e.g. choose between a line with a small triangle or an arrow to represent *isA*-relationship in ER diagrams. Other examples of notational adaptability which do not affect ontologies is a change in the notation still using the same presentation style (e.g. DFD of DeMarco [5] to Yourdon [26], or use different presentation styles (e.g. a graphical representation of entities instead of a list).

2.1.3 Process Adaptability

Process models capture a set of partially ordered steps intended to produce the desired product [16]. In creative domains, like ISD, the entire process cannot be fully predefined in advance. Instead, the process should be defined for small, well-known fragments and adapted and extended while process knowledge increases [9]. Organizations and individuals continuously experiment with the most appropriate way of working even if a consensus on the ontology and notation has been reached. Hence, process adaptations most often neither change the concepts nor the representations but merely influences how processes are carried out. Examples of process adaptation are defining new strategies for integrating two ER schemas, or changing the order in which the entities and relationships should be identified from a use case description and added into the ER diagram.

2.2 Integrated Adaptability

The majority of adaptations cannot be done adequately by changing one aspect of a method only. In the diamond model, these adaptation possibilities are illustrated by the relationships between ontology, notation and process (cf. Fig. 1). Depending on the view taken on metamodeling, several starting points are possible: In the case of ontology-centered metamodeling, integrated adaptability starts by defining the underlying conceptual structure, then relating notations to the concepts, and finally defining processes for manipulating these concepts (arrow 5 and 6, Sect. 2.2.1). In a notation-centered metamodeling scenario, the type of documents built with modeling techniques determine the ontologies used and process followed (arrow 7, Sect.

2.2.2). In a process-centered metamodeling scenario, changes in a process can suggest new or changed concepts and their representation (arrow 4, see Sect. 2.2.3).

2.2.1 Ontology-driven Adaptability

Changes in an ontology can reflect either or both to notations and processes (arrow 5 and 6 in Fig. 1). From the notational perspective, new or changed ontological concepts result in adding, changing or purging representations. For example, a line with a small triangle may be required to represent a newly introduced *isA*-relationship in ER diagram. Another example is a definition of diagram representation for use case models.

Modifications in an ontology always induce changes in a process because an ontology affects the overall way of working. For example, if the ontology for use case modeling has been added, a set of elementary process steps for manipulating these concepts (e.g. creation, modification, deletion of actors and use cases, relation of actors to use cases etc.) is required. Note that each ontological construct can only be manipulated via an associated notational element. Therefore, process steps are normally defined through changes in a notational system (illustrated by a dotted line in Fig. 1).

2.2.2 Notation-driven Adaptability

Notation oriented metamodeling can lead to changed process steps (arrow 7 in Figure 1). These steps deal with notational aspects of representations (cosmetics) without affecting the underlying conceptual structure. Examples of such are diagonalizing a matrix, or sorting entities in a list. In the latter case, adding an entity list together with ER-diagram necessitates that the process model defines when and how entity lists are created and sorted. As a result, the readability of the representation is improved and user can infer more information from it although the representation itself provides equal information. Although some changes could also lead to change the ontology we think it is not common nor desirable scenario for method adaptation.

2.2.3 Process-driven Adaptability

Process adaptation is typically triggered by changes in an ontology or notation, since processes act on top of notational (and indirectly upon ontological) primitives. In other words, a process definition requires the existence of a notation (and an ontology). However, process model changes can conversely also call for adaptations in the other corners of the diamond. We believe that such adaptations mainly affect notations without changing the underlying ontology. For example, consider a process fragment "Simplify ER schema" which guides the user in checking a set of entities whether they can be omitted for simplifying the ER schema. If the set of entities to be checked is large, one would like to distinguish checked entities from non-checked ones. This calls for the definition of special symbols for checked entities.

3 Support for Adaptability

We now present two adaptable environments that provide partial solutions to the adaptability problems presented in section 2. MetaEdit+ [11] is a notation and ontology-oriented meta-CASE tool, which provides tools for specifying an ontology and a related notation and generating automatically (on the fly) a set of modeling tools to support the method. PRIME[1] [18] focuses on process-driven metamodeling. It allows to specify processes, to enact them, and to reflect the process enactment directly in the behavior of the modeling tools.

[1] PRIME: *PR*ocess *I*ntegrated *M*odeling *E*nvironments

3.1 MetaEdit+

MetaEdit+ is a multi-user CASE environment in which each MetaEdit+ client contains a set of modeling tool for ISD and a set of metamodeling tools for making the adaptation.

At the IS modeling side, MetaEdit+ provides three modeling editor types (Diagram Editor, Matrix Editor, Table Editor) which are adapted based on the metamodel. Hence, diagram editor can support both structured methods and object-oriented methods. Each editor type contain a set of predefined operations which are independent from any ontology or notation. Examples of these are scrolling and zooming in Diagram Editor, diagonalising in Matrix Editor for and sorting in Table Editor. Each of the editor related operations are predefined and no adaptation among them is possible. Other parts of the process are dependent on the method followed in an editor.

At the adaptation side, MetaEdit+ provides metamodeling tools for defining concepts and associated notations, which can be graphical, matrix and tabular ones. The metamodeling tools have been applied to model tens of methods in terms of their ontology and notation. An ontology is defined by using the GOPRR [11] (Graph, Object, Property, Role and Relationship[2]) metamodeling formalism (see Fig. 2). Graph specifies a technique (graph type), Object, Role and Relationship specify elements of a graph type and Property specifies property types to describe these elements as well as graph types. The structure of how these elements are bound together is managed in a graph type. In addition, GOPRR supports explosions and decompositions (specifies what object types can be linked to graph types) and links between property and object types. Each GOPRR concept can contain a graphical representation which defines the appearance of the concept.

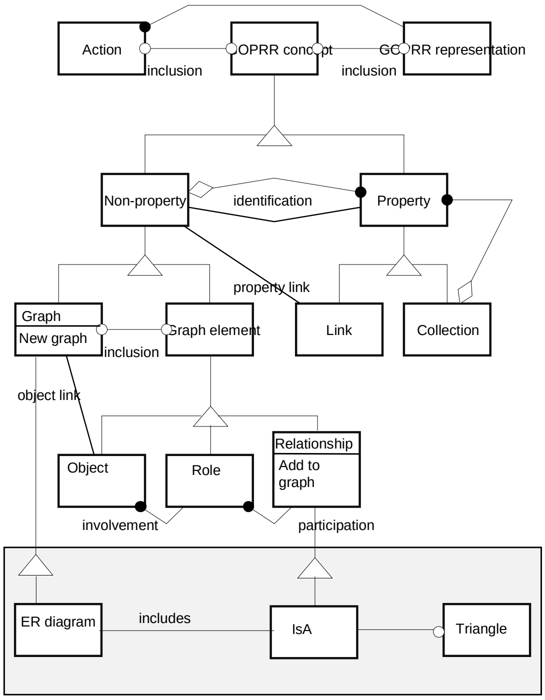


Fig. 2: GOPRR meta-metamodel

Pure notational adaptation is straightforward in MetaEdit+ thanks to the separation between conceptual and notational elements. A user can change between graphical, matrix and tabular notations which all are based on the same conceptual data. For example, an ER diagram is easily changed to a form of entity list by just opening a Table Editor for an existing graphical diagram. Modification of notational elements are done by using metamodeling tools. For example, when changing the symbol of 'isA' we need to select it in Relationship tool and update the symbol by using a Symbol Editor. The changes are seen immediately when opening a model containing 'isA' relationships. Related to the process part, no modifications is possible based on notational changes. In other words, MetaEdit+ provides a set of fixed elementary operations for manipulating notations (e.g. size, color, etc).

Next we discuss how to create a new relationship type 'isA' between entity types in ER model and a new representation for it (notational adaptation affected by ontological changes). Firstly, we create 'isA' relationship type by using Relationship tool. Secondly, we add 'isA' to the existing graph type (i.e. ER diagram) and make a new binding type: 'isA' involves two role types,

---

[2] In the following, instances of types are in lowercase, e.g. *object*, types are appended with 'type', e.g. *object type*, and metatypes are initially capitalised, e.g. *Object*.

a ‘super part’ and a ‘sub part’, both of which is participated by an ‘entity type’. Also additional constraints can be added here, such as possibility to create n-ary relationships. Thirdly, we attach a triangle symbol for ‘isA’. After these steps we can generate modifications to be used in editors.

The result of the adaptation is illustrated in the grayed part of Figure 2. ‘IsA -relationship is part of ER diagram and it has a representation definition. The fixed process structure is illustrated in the figure through the operations defined at the GOPRR. GOPRR provides predefined elementary operations, such create and remove, which in case of relationships include adding a relationship into a graph. This operation is possible for all relationships independently of the specific ontology and notation.

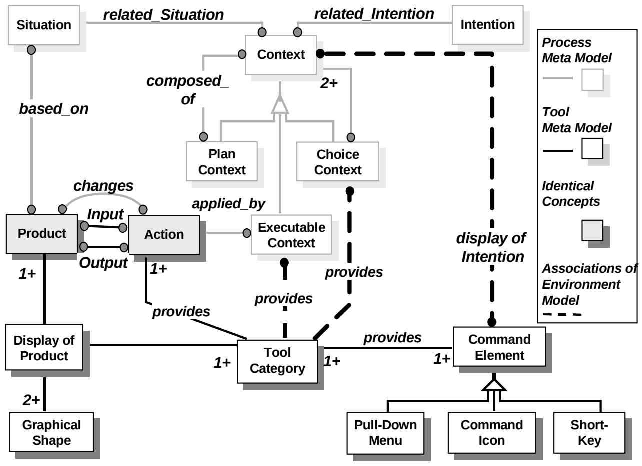
To illustrate how to add a whole ontology and associated notation we have selected use case diagrams. We first define a graph type called a ‘use case diagram’, and its concepts as object types (‘actor’ and ‘use case’), relationship types (‘communication’, ‘extends’ and ‘uses’), and role types (‘receives’, ‘extends/uses’, and ‘is extended/used’). Each type is modeled as separate and reusable fragment. Each of them can be identified and described with property types as follows: the property types ‘name’ and ‘description’ are attached to the graph type ‘use case’ and the object types ‘actor’ and ‘use case’. For each property type there exist an entry field in a dialogue, which appears to be filled when creating or editing instances of these types. When composing the use case technique object, relationship, and role types need to be bound together (see isA example above). To allow hierarchical models we define a decomposition from the ‘use case’ to the ‘use case diagram’. After we have defined the ontology we attach notational elements for each type, a help description, and consistency checking reports. An appropriate “human” symbol is attached to the object type ‘actor’ and an ellipse is attached to the object type ‘use case’. Role types are presented with various types of lines (black line for communication, grayed line for others) and line-end symbols (black arrow for communication, grayed arrow for ‘is extended/used’). Relationship types do not have symbols in the use case diagram.

3.2 PRIME

PRIME aims at incorporating fine-grained, adaptable process guidance into CASE environments, thereby forming a *PR*ocess-*I*ntegrated *M*odeling *E*nvironment. PRIME is based on an environment metamodel for homogeneously modeling both processes and tools and a generic architecture for process-integrated tools which adapts tool behavior to the definition in the environment model. PRIME is implemented as an object-oriented framework and has so far been specialized to develop tools for two environments: PRO-ART 2.0, a requirements traceability environment [17], and TECHMOD, an chemical engineering environment.

In contrast to traditional process-centered environments like SPADE [2] or ProcessWeaver [7], PRIME-based tools (1) are process-sensitive and (2) able to request the enactment of process fragments from the process engine. Process-sensitive tools reflect process model enactment in their user interface by dynamically adapting access to and highlighting those products and operations which are allowed and useful in the current enactment state according to the process definition. Requesting enactment means that the tools do not only offer at their user interface their elementary operations (like creation, modification and deletion of products), but also allow the invocation of more complex process model fragments which are applicable in the current situation.

The central part of PRIME’s environment meta-metamodel (upper half of figure 4) is the *Context* [19], [20]. A context represents a process fragment and relates a certain *Intention* the user wants to achieve with a *Situation* in which the intention is applicable. A situation describes a constellation of the products under development. How a context is performed is detailed by specializing the context into three distinct categories: (1) an executable context operates through an action on the products under development and represents an atomic process step; (2) a plan context is composed of a set of contexts and defines a control structure among the contexts of this set; (3) a choice context describes a decision point within a process fragment and is defined by the set of applicable alternatives contexts. If a choice context is activated, the user can decide among these alternatives.



The lower part of the environment meta-model offers concepts for modeling the basic capabilities of a tool category. Tools are not only modeled in terms of the actions they provide and the products they operate on, but also by which graphical shapes (i.e. notational elements) an editor is able to display a product (i.e. an ontological construct). The interaction capabilities are modeled by the command elements (i.e. menus, command icons, short keys) a tool offers.

By interrelating the process model and the tool model through a small number of relationships an integrated environment model is formed. These relationships assign executable and choice contexts to certain tool categories. PRIME-based tools interpret the definitions in the environment model and thereby reflect process adaptations in an accordingly changed tool behavior, e.g. offer new process fragments as new command in their user interface (for details see [18]).

We now illustrate by a brief example how PRIME supports the process adaptability scenarios sketched in section 2. *Pure process adaptability* involves the definition of new or modification of existing process fragments which solely relies on existing ontologies and notations. A large-grained example for pure process adaptability is to define a new plan context which guides the user during schema integration of two ER models. The plan context could specify that the user should first identify synonyms and homonyms in the two ER models, then create a new diagram, and finally transfer the entities and relationships from the old diagram to the new diagram with appropriate identifications of identical entities and renamings of conflicting entities. These substeps are also modeled as contexts, e.g. a choice context in which the user can decide if he wants to merge two entities with the same name to one or to rename them differently. At the lowest decomposition level, executable contexts are defined and operationalized by actions of the ER editor. The new plan context can be made visible in the ER editor by relating it to a choice context of the ER editor and assigning a control element (e.g. menu point) of the ER editor to the new plan context.

Notation-driven process adaptability is primarily determined by the support for notation adaptation offered by PRIME. Currently, limited flexibility is provided in that the method engineer can choose among a predefined set of graphical primitives for a certain ontological construct. This enables for example the method engineer to define a process fragment when the ER editor should display entities by a rounded box instead of using rectangular boxes.

Similarly, the method engineer is empowered to select between a set of predefined ontological elements provided by the tools. For example, he could hide the concept of „IsA“ relationship in the ER editor from the IS developer by removing the corresponding association between the product „IsA-Link“ and the tool category „ER Editor“. At the process level, this would require to remove all existing process fragments which manipulate the isA concept. Hence, process support can be adapted according to the subset of ontological constructs actually used in a specific environment.

3.3 Summary and Comparison

Table 1 compares the adaptability strengths of MetaEdit+ and PRIME with respect to the adaptation scenarios applied in section 2. The support provided in a certain scenario is classified into full flexibility (+), limited flexibility, i.e. choice among a predefined set of alternatives (O), and no flexibility or consideration of this aspect (–). Of specific interest are the rows 4 - 7, which deal with integrated adaptability, i.e. changes in one diamond node triggered by a change in another node. Here the adaptability support has to consider the diamond node from which a change is originated as well as the diamond node which is affected by the change[3].

Adaptation scenarios	Granularity	Example	Support provided by	
			ME+	PRIME
1. Ontological adaptability	Technique	Define use case ontology	+	–
	Concept	Add inheritance type between entity types	+	O
2 .Notational adaptability	Notation	Create list of entities	+	–
	Not. Element	Change a symbol for an isA relationship type	+	O
3. Process adaptability	Large	Define the task for schema integration	O	+
	Small	Change order for creating entities and relationships	–	+
4. Notational adaptability affected by process changes	Notation	Process fragment for deriving ER model from a use case model has been defined ==> Create use case description template	(+)	–
	Not. Element	Process fragment for systematically checking (a large number of) entities has been defined ==> Create symbols for checked entities	(+)	O
5. Notational adaptability affected by ontological changes	Notation	Use case ontology has been defined ==> Add a use case diagram	+	–
	Notational Element	IsA concept has been added to ER ontology ==> Add representation for an isA relationship	+	(O)
6. Process adaptability affected by ontological and notational changes	Large	Use case ontology and representation has been defined ==> Define when and how use cases diagrams are related to ER diagram	O	(+)
	Small	IsA concept and representation has been defined ==> Define how to check other models when an isA link between entities is created	–	(+)
7. Process adaptability affected by notational changes	Large	List representation for ER models has been added ==> Define when entity lists are created	O	(+)
	Small	List repr. has been added ==> Define when to sort entities	–	(+)

Table 1: Adaptability scenarios and support provided by ME+ and PRIME.

To summarize, MetaEdit+ provides high levels of ontological and notational adaptability. Currently it provides no process support, i.e. the actions offered are largely driven by static information about the generic behavior of ontological constructs and presentation styles and independent on any process definition. There is no notion of composing elementary operations to more complex process fragments, or to restrict availability of operations on certain modeling objects depending on the situation. Furthermore, regarding the tool support, each editor type contains a fixed tool functionality (e.g., zooming, scrolling, diagonalizing, or sorting) relevant for the editor type in concern and independent from methods. This strategy has its advantages such as easiness to manage method-tool companionship. Hence, the main limitations are seen in the process adaptability aspects in figure 1, which can not be supported in MetaEdit+.

PRIME is best suited for supporting pure process adaptability, i.e. if there is no need to define new ontological constructs or notational elements. Hence, the focus of metamodeling in

[3] The adaptability classification is put into parenthesis if the *triggered* change is in principle supported, but in practice the adaptation scenario does not occur since the *triggering* adaptability is not adequately supported in the environment. For example, in row 6 it would be no problem in PRIME to define new process fragments for systematically treating a new defined isA concept once the basic operations for creating, modifying, deleting and representing it are available in the ER editor. However, since PRIME does not allow to extend an existing ontology (on a metamodeling basis) this scenario cannot be fully exploited.

PRIME is to define and adapt processes in a flexible manner and to incorporate them in the tools on top of existing tool support for ontologies and notations used in the environment. Furthermore, PRIME allows to choose the representation of a concept among a set of predefined shapes, e.g. to change the appearance of an entity from a rectangular box to a rounded one. Currently the set of basic operations on an ontological concept as well as its appearance choices are to certain degree fixed in the tool supporting these concepts. Hence, process adaptation originated by the introduction of new ontological or notational elements is not supported in PRIME.

4 A Comprehensive Metamodel-Based Adaptability Approach

An interesting observation drawn from table 1 is that the adaptability strengths of both environments are clearly complementary. In this section we combine the adaptability ideas underlying MetaEdit+ and PRIME in order to address all corners of the method adaptability diamond and their relationships (Sect. 4.1). An example for enhanced adaptability which separate tools can not adequately support is sketched in Sect. 4.2.

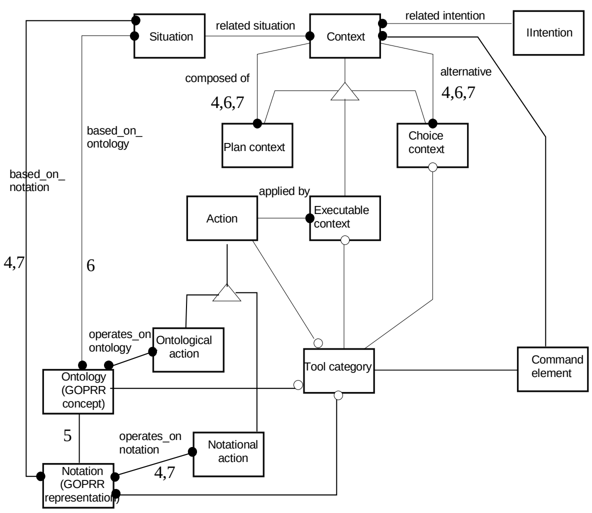
4.1 Integrated Metamodels

The key strategy for comprehensive adaptability is to integrate the complementary metamodels and underlying adaptability mechanisms of MetaEdit+ and PRIME. Figure 4 gives a (simplified) overview of the integrated metamodel. The numbers in the figure refer to the relationships of the diamond model supported by the integrated metamodel.

The integration strategy is to replace the product part of PRIME's metamodel by the more powerful GOPRR concepts. One the one hand, this enables the use of existing support for adaptability: Ontological constructs and notational primitives supported by MetaEdit+ and process adaptability supported by PRIME. On the other hand, to enable integrated adaptability, process fragments are defined on top of the GOPRR based elementary ontological and notational actions with PRIME's process modeling concepts.

More precisely, complex product constellations both referring to the conceptual structure and to their representation are defined using the *Situation* concept. Furthermore, each elementary GOPRR action is „wrapped„ by an *Executable Context*. Based on these definitions more complex process fragments can be modeled, either as choice contexts or plan contexts. Thus, on the one hand access to GOPRR based actions can be made dependent on the current process context (by defining choice contexts on top of them). On the other hand, the definition of plan contexts serves for composing sequences of process steps. (number 6 in Fig 5.).

This integration leads to a comprehensive metamodel which covers the ontology, notation, and process aspect of a method. In contrast to the situation described in section 3, tools now are not only able to adapt their behavior according to the GOPRR based tool models which specify the products a tool manipulates and their representations but also to the processes defined in the contextual process models.



4.2 Examples of Integration

We now can view the integrated adaptation scenarios in the light of the integrated metamodeling formalism and the associated generic mechanisms. We illustrate through a short example the benefits gained by combining MetaEdit+'s ontology and notation adaptability and PRIME's process adaptability. Consider an ISD organization which uses the traditional ER approach. The development of ER models is supported in the CASE environment by an ER editor. After a while the ISD organization decides to use the Extended ER modeling. This imposes new requirements on the ER tool affecting all possible adaptation scenarios. Accordingly, in an integrated environment the adaptability can be approached either from ontology, notation or process point of view. Since adaptations are already partially supported in current tools the illustration is based on the adaptation scenarios which could not be supported with separate metamodels and thus with separate tools (numbered 4,6,7 in Fig. 1 and Table 1).

Ontology driven adaptation deals with adding a new concept for expressing specialization of entities, namely the concept of isA-relationship. This modification leads to enhance the notation, namely to add an arrow for representing isA-relationships, and the definition of necessary actions (e.g. creating, delete, modify, etc.) related to that concept. Next we can define new process situations involving the new isA concept and elementary process fragments, i.e. executable contexts. Accordingly, the process model then provides these executable contexts for the isA-relationship. More complex and larger-grained process fragments are then modeled as choice and plan contexts, which, e.g., prohibit the IS developer from defining circular isA-relationships

Process driven adaptation emphasizes that we first define the process which is required to identify, create and refine isA relationships depending on the context and current modeling situation. For example, a guidance chunk, which models the possible choices for the refinement of an existing entity, must be augmented by an additional alternative, namely creating a sub entity and establishing an isA-Relationship between the old entity and the newly created entity. In our adaptation scenario pure process adaptability is not enough since process changes need also to be reflected to ontological and notational constructs. This can now be achieved as described in the previous paragraph.

Notation driven adaptation in this case deals with distinguishing the actions which could guide when and how the new concept should be represented, e.g. when it is appropriate to display an ER model in diagram representation or matrix representation. Elementary actions are dependent on the editor type, e.g. in diagram editor isA relationships are shown as lines whereas in matrix editor they are shown through cells between two or more entities at axis. Notational actions are also dependent on the method supported. In a case of isA-relationships and its representation, the process model could prescribe that in the diagram editor the sub-entity types should for example appear below the supertypes and at the same horizontal level.

5 Conclusions and Future Work

In a companion paper we have introduced an adaptability framework [10] which distinguishes ontology, process, and notation. In this paper we have focused in more detail on CASE environment adaptability involving two or three of these aspects at a time. Based on the observation that current environments only focus on certain aspects while neglecting the others, we have then compared two environments with complementary adaptability strengths: MetaEdit+ and PRIME. While MetaEdit+ provides high degrees of ontological and notational adaptability, PRIME offers very flexible means for incorporating fine-grained, adaptable process guidance into tool environments.

To combine these strengths synergistically, we have presented an integration of both environments' meta-metamodels. The key idea is to re-use MetaEdit's product part as a basis for providing elementary actions. On top of these elementary actions we can define more complex process guidance using PRIME's process modeling concepts. As a result, we can define the context of process fragments more precisely in terms of product constellations which refer to the current state of both ontology and notation. This is demonstrated through a small example illustrating how such an integrated metamodeling formalism paves the way to a comprehensive adaptability solution.

The integration of metamodels is only one step towards a fully adaptable CASE environment. Future work has to concentrate on how to integrate the associated adaptability mechanisms provided by MetaEdit+ and PRIME. A step forward is to analyze how the metamodels can be stored and retrieved efficiently during tool execution. Furthermore, we need also to study necessary mechanisms for maintaining the integration between different types of metamodels (i.e. product and process). This means identification of metamodeling guidelines when adaptation requirements raise from one aspect of adaptation. Currently, we are examining the interdependencies of these mechanisms at the architectural level, thus allowing for an assessment of the potentials and difficulties of a technical integration of both environments.

Acknowledgement. This work was partly supported by the European Commission via ESPRIT Long Term Research project 21.903 (CREWS).

6 References

[1] Aalto, J.-M. (1993) Experiences on Applying OMT to Large Scale Systems. In: Proceedings of the Seminar on Conceptual Modeling and Object-Oriented Programming, (Eds. A. Lehtola, J., Jokiniemi), Finnish Artificial Intelligence Society, pp. 39-47.

[2] Bandinelli, S., Fuggetta, A. and Ghezzi, C. (1993) Software Process Model Evolution in the SPADE Environment. IEEE TSE, 19(12), pp. 1128-1144.

[3] Booch, G., Jacobson, I., and Rumbaugh, J. (1997) Unified Modeling Language: The summary. Rational Software Corporation.

[4] Davis, G. (1982) Strategies for information requirements determination, IBM Systems Journal, 21(1), pp. 4-30.

[5] DeMarco, T., (1979) *Structured Analysis and Systems Specification*. Englewood Cliffs, N.J., Prentice-Hall.

[6] Euromethod (1994) Euromethod Architecture. Euromethod project deliverable work package 3.

[7] Fernström, Ch. (1993) Process WEAVER: Adding Process Support to UNIX. In: Proc. 2nd Intl. Conf. on Software Processes, Los Alamitos, CA, USA, pp 12-26.

[8] Fitzgerald, B., (1995) The use of system development methods: a survey. Paper ref 9/95, University College Cork.

[9] Jarke, M., Pohl, K., Rolland, C., Schmidt, J.-R. (1994) Experience-based Method Evaluation and Improvement: A Process Modeling Approach. In: IFIP WG 8.1 Conference CRIS '94, Maastricht, The Netherlands.

[10] Jarke, M., Pohl, K., Weidenhaupt, K., Lyytinen, K., Marttiin, P., Tolvanen, J.-P., Papazoglou, M., (1997) Meta modeling: A formal basis for interoperability and adaptability. In: *Information Systems Interoperability* (B. Krämer, M. Papazoglou), John Wiley RSP.

[11] Kelly, S., Lyytinen, K. and Rossi, M. (1996) METAEDIT+ — A Fully Configurable Multi-User and Multi-Tool CASE and CAME Environment. In: Advanced Information Systems Engineering, LNCS#1080, Springer, pp. 1-21.

[12] Marttiin, P. (1994) A Comparative Review of CASE Shells: a preliminary framework and research outcomes. Information and Management, 25, pp. 11-31.

[13] Necco, C.R., Gordon, C.L. and Tsai, N.W. (1987) Systems Analysis and Design: Current Practices. MIS Quarterly, December, pp. 461-475.

[14] Nissen, H., Jeusfeld, M., Jarke, M., Zemanek, G. and Huber, H. (1996) Managing multiple requirements perspectives with metamodels. IEEE Software, March, pp. 37-48.

[15] Olle, T.W., Hagelstein, J., MacDonald, I.G., Rolland, C., Sol, H.G., Van Assche, F.J.M. and Verrijn-Stuart, A.A. (1991) Information Systems Methodologies — A framework for understanding, Addison-Wesley Publishing Company, Wokingham, England.

[16] Pohl, K. (1996a) Process-Centered Requirements Engineering. Wiley&Sons, New York.

[17] Pohl, K. (1996b) PRO-ART: Enabling Requirements Pre-Traceability. In: Proceeding of the 2nd International Conference on Requirements Engineering, Colorado Springs, Colorado, USA.

[18] Pohl, K., Weidenhaupt, K. (1997) A Contextual Approach for Process-Integrated Tools. In: Proc. ESEC/FSE '97, Zurich, Switzerland, Sept. 23-25, pp. 176-192.

[19] Rolland, C., Souveyet, C., Moreno, M. (1995) An approach for defining ways-of-working, *Information Systems*, 20(4), pp.337-359.

[20] Russo, N., Wynekoop, J. and Waltz, D. (1994) The use and adaptation of system development methodologies. In: Procs. of International Conference of IRMA, Atlanta, May 21-24, pp.

[21] Schefström D. (1993) System Development Environments: Contemporary Concepts in Tool Integration and Frameworks. (Eds. Schefström D., and G. van den Broek), Wiley&Sons, Chichester.

[22] Smolander, K., Lyytinen, K., Tahvanainen, V.-P. and Marttiin P. (1991) MetaEdit - A flexible graphical environment for methodology modelling. In: Advanced Information Systems Engineering, (Eds. R. Andersen, J. Bubenko and A. Sølvberg), LNCS #498, Springer-Verlag, pp. 168-193.

[23] Thomas I. and Nejmeh B.A. (1992) Definitions of Tool Integration for Environments. IEEE Software, March, pp. 29-35.
[24] Wand, Y. (1996) Ontology as a foundation for meta-modelling and method engineering. Information and Software Technology, 38, pp. 281-287.
[25] Wijers, G. (1991) Modelling Support in Information Systems Development. Ph.D. dissertation, Thesis publishers, Amsterdam.
[26] Yourdon, E., (1989) *Modern Structured Analysis*. Prentice-Hall, Englewood Cliffs, New Jersey.