Towards a Framework for Model Migration

Werner Esswein, Andreas Gehlert, and Grit Seiffert

University of Technology Dresden, Dresden, 01069, Germany
{esswein,gehlert}@wise.wiwi.tu-dresden.de
grit.seiffert@mailbox.tu-dresden.de
Phone: ++49 351 463 36304, Fax: ++49 351 463 37203

Abstract. Model migration is needed to transform a model from one modelling language to another to reuse these models and thus, to reduce the costs of building new models. This topic has received much attention mainly because of the OMG's MDA initiative. However, these approaches are tailored to transform design models. In this paper, we argue that this approach cannot be applied to the field of domain modelling. Therefore, we propose a theoretical framework for model migration, which extends the existing approaches adding a role and a process model to satisfy the requirements of model migrations in a material problem domain.

Keywords: model migration, domain modelling, modelling method

1 Introduction

Modelling is an important technique to gather insights into specific problem domains. Models are created by specific – mostly visual – modelling languages. These modelling languages are suited to analyse a specific aspect and/or view of a problem domain (see [1], p. 401, [2], p. 22f). Thus, different modelling languages have been created using different concepts suited for these views and aspects. The number of modelling languages has further been increased by the method engineering discipline (see [3], [2] and [4]), which aims to propose modelling languages especially suitable for a specific project. Because of conceptual discrepancies interchanging of models between different modelling languages is non trivial.

One of the prerequisites for reusing models is, that the modelling language of the original model is the same as the language used for a specific project at hand. If this is not the case, the original model has to be transformed into the new modelling language. If the modelling languages involved in such a transformation have different concepts (not only a different graphical representation), this process is called model *migration*[1].

Many solutions to the migration problem have been presented ranging from an ad hoc comparison of meta models for a specific purpose (see [6], p. 58) to complete methodologies for model migration (see [6], [5][2]). These approaches are

[1] This process is also known as model transformation (see [5], p. 2-7).
[2] For an overview of the approaches in the MDA field, see [7].

A. Persson and J. Stirna (Eds.): CAiSE 2004, LNCS 3084, pp. 463–476, 2004.

tailored for migrating design models only. Design models regard a mathematical or formal domain. However, in a material problem domain a complete model migration has to respect syntactic, semantic and especially pragmatic aspects of models (see subsection 2.1). Current approaches to model migration do not account for the pragmatic aspects in a material problem domain.

Therefore, we propose in section 2 of this paper a theoretical framework for model migration, which extends the existing approaches to satisfy the requirements of a model migration in a material problem domain. This framework consists of a role model (subsection 2.2), a process model and an – interchangeable – meta model based migration approach (section 3). Furthermore, we provide an example demonstrating the application of the framework (subsection 3.3). The paper finishes with a summary of the results and states the remaining problems.

2 Theoretical Foundations of Model Migration

A *model* is characterised by three properties. First, models are representations of (natural or artificial) originals (problem domain). Second, not the complete problem domain is represented in such a model but only the relevant parts of it. Third, each model has a pragmatic relativization, which means it is only valid for a special purpose, at a certain time and for a special (group of) user(s) (see [8], p. 131ff).

In this paper we use the term model to refer to *domain models* as visual models for the purpose of information system development in its broadest sence. The problem domain is – in contrast to implementation models for instance – a material not mathematical or logical domain (see subsection 2.1 for that differentiation).

Any model serves as a problem definition. Problems are per definition not well-structured, highly subjective and individual and thus, not objectively well formed (see [9], p. 2). Consequently, domain modelling is always concerned with structuring the real world not with merely mapping it. Domain models which fulfil this definition are for example ER models (see [10]), EPC models (see [11]) and UML models (see [12], p. 3-34ff).

2.1 Semantics and Pragmatics in Models

After the clarification what we understand by the term model, we analyse the elements to which semantics and pragmatics in models are assigned. These elements are of special interest for reusing and migrating models (see [6], p. 23f; [13]).

The modelling language communicates important pars of the semantics of the model. In general, three aspects of languages are important. First, the *syntax* describes the language expressions and their lawful combinations, second, the *semantics* maps these expressions to a semantic domain (see [14]) and third, the *pragmatics* describes the meaning of an expression for the individual who uses it.

Two stances of semantics must be separated. First, semantics understood as rules for automated interpretation as it is used implicitly in the MDA approach

and, second, semantics as interpretation of real world constructs and their representation in language artifacts (see [13]). The later terminology is used in the field of domain modelling.

Consequently, domain models always represent concepts of real world entities. This conceptualisation and interpretation of the real world cannot be formalised because it is determined in a social process (see [15], p. 127).

As stated before, a model is always represented by a specific modelling language. Furthermore, it is constructed using an implicit or explicit process model. The models of the modelling language and the process model are called *meta models* (MM). To distinguish between these meta models we name them meta model of the modelling language[3] (MM_L) describing the syntax of M and meta process model (MM_P) describing the modelling process of M. Both meta models form a modelling method (see [3], p. 11). Figure 1 summarises these aspects.

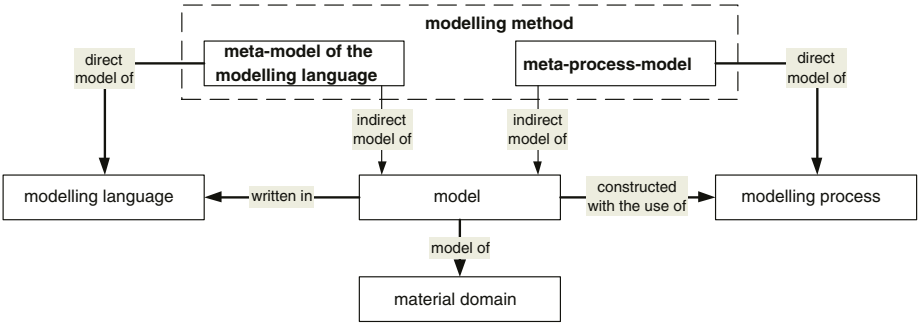


Fig. 1. The role of meta models (according to [16], p. 958, figure 2)

To specify the semantics of a model the modelling language and the modelling process must be highlighted. The semantics of a model element, $sem(E)$, is primarily described with the help of the concepts of both meta models (C_{MM_L} and C_{MM_P}). This aspect is called *abstract semantics* ($sem(E_a) = sem(C_{MM_L}) \cup sem(C_{MM_P})$). For example, an entity type `order` has semantics because it was modelled as entity type. Thus, it is seen as a concept of the real world and not a relation between concepts in the real world (see [10], p. 11f). Furthermore, semantics is added because this entity type was created within a concrete modelling process.

Additionally, each model element adds a specific meaning, which we refer to as *concrete semantics* ($sem(E_c)$). For example, the entity type `order` itself carries a meaning (e. g. customer-order). Consequently, the semantics of a model element can be defined as union of the abstract and concrete semantics of this element ($sem(E) = sem(E_c) \cup sem(C_{MM_L}) \cup sem(C_{MM_P})$).

In line with recent research in the field of visual modelling, the layout of modelling graphs can also communicate semantics (see [13], [17]). For instance,

[3] Often referred to as meta model of the abstract syntax (see [14], p. 5).

a grouping of processes could indicate that they belong to each other. According to the method engineering discipline we regard this aspect to belong to the modelling language and is thus described in its meta model MM_L (see [1], p. 405). So, the semantics communicated by the layout of the graph is included in $sem(C_{MM_L})$.

Individuals interpret the semantics of a model in different ways. This is the pragmatic aspect of a model. So, individuals assign a concrete pragmatics to any model element ($prag(E_c)$).

Moreover, the same can be said about the modelling methods and their description MM_L and MM_P. Individuals form their own interpretation of the meta model's concepts C_{MM_L}, C_{MM_P} ($prag(C_{MM_L})$ and $prag(C_{MM_P})$). The pragmatics of any model element can now be described as $prag(E) = prag(E_c) \cup prag(C_{MM_L}) \cup prag(C_{MM_P})$.

These pragmatic aspects are assigned to a model by the model creators during the interactive modelling process. Because the pragmatics is manifested in the relation between the language expressions and the individuals who use them, it is not explicit in any model[4].

Models are used as an instrument for solving problems in different domains. Therefore, these models are usually created by domain experts rather than modelling experts. These domain experts may implement their knowledge in different ways in models because of their potential inexperience in modelling. Consequently, the pragmatic aspect of models needs to be addressed to migrate domain models.

In contrast, meta models are used for describing models. They are created by modelling experts only. Thus, the differences in the interpretation of different meta models will be minimal. Therefore, the pragmatic aspect of meta models will be neglected at this point.

To sum up the analysis so far: The information assigned to a model has not only semantic but particularly pragmatic aspects. These pragmatic aspects are not explicit in any model. Therefore, models are interpreted differently by different individuals. Consequently, we need to develop an organisational framework to examine under which circumstances reuse and migration of domain models is feasible. This will be the focus of the next subsection.

2.2 The Role Model

To take the pragmatic aspects of models into acount we must further analyse the roles involved in the modelling and the model migration process. The subsequent argumentation is summarised in figure 2.

[4] Important prerequisites to reduce the pragmatic influence in models are a rigorous definition of the modelling method, the suitability of this method for the problem and the rigorous use of this method. Many approaches have been developed to fulfil these requirements such as the Guidelines of Modelling (see [18]) and the use of ontologies (see [19] as one example). However, since pragmatics describes the usage of language expressions by individuals, each modeller will assign pragmatic aspects to models. Therefore, it will not be possible to completely rule out these pragmatic aspects.

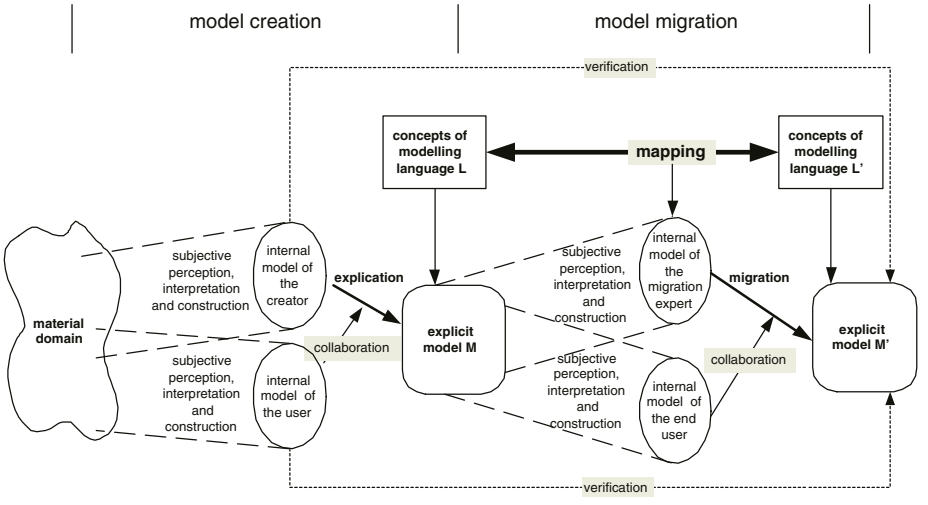


Fig. 2. Different roles in the modelling and migration process (see [20], p. 61)

To create a model at least two roles must be separated. On the one hand there is the creator of the model and on the other hand there is the one who uses the model to solve a specific problem. Both roles can be implemented by groups of different individuals or by a single person only.

During the modelling process people involved create automatically an internal model of the problem domain and explicate this model (external model) step by step in interaction with all other individuals. The process ends if everybody has reached a consensus about the internal and external models. Consequently, at the end each individual involved in this modelling process has the same understanding not only of the semantics but particularly of the pragmatics of the resulting model.

If the model M is to be transformed from its modelling language L to the modelling language L', two new roles – the migration expert and the end-user – are needed. The migration expert takes M as input and creates automatically an internal model of it. With the help of the modelling languages L and L' and the end user he or she creates the model M'.

If the migration expert did not carry the role of the user or the creator of the model M, a transformation of all information into the new modelling language is not possible, because of the implicit pragmatics. Consequently, a verification of the resulting model M' by its users or creators is necessary. This insight leads us to the initial migration process (see figure 3).

This process takes a model M created in a modelling language L as input and produces a model M' of a modelling language L' ($L \neq L'$) as output. We expect the resulting model M' to have the following properties:

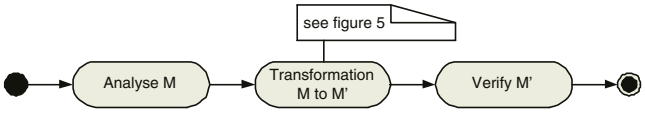


Fig. 3. Initial process model

1. M' is syntactically correct against L',
2. M' has the same semantics as M and
3. M' has the same pragmatics as M for the creator, user, migration expert and the end-user.

Additionally, the *information loss* – the information lost because of the semantic weakness of the destination language – and the *information deficit* – the information added to the source model because of the semantic richness of the destination language – must be determined explicitly.

To enhance the effectiveness of the migration process the migration expert should work independently from the creator and user of the initial model M. Furthermore, the process should be automated as far as possible.

In the next subsection we show how the transformation process itself can be unfolded.

2.3 Migration Costs

To analyse the transformation process in detail we first concentrate on different techniques for model migration. Second, we analyse the costs of these techniques. Third, we develop our own framework on that basis.

There are two different ways for migrating a model M created with a modelling language L to a model M' in a modelling language L'.

1. The migration can be founded on the model M only. This means that the language descriptions MM_L and $MM_{L'}$ are not used for the transformation.
2. The migration can also be done by using the meta models. That means the migration expert must determine the concepts of MM_L used in M and must create a mapping of these concepts to $MM_{L'}$. After this mapping the concrete model elements of M can be transformed.

The first approach describes a remodelling of the problem domain but disregards the semantics of the meta model concepts. Therefore, we follow the second approach.

To analyse this approach in further detail the costs of the transformation process must be considered. The converter that implements this transformation determines these costs. This converter has three responsibilities:

1. Mapping of the concepts C_{MM_L} to $C_{MM_{L'}}$.
2. Transformation of all instances of the concepts C_{MM_L} into their counterparts ($I_{C_{MM_L}}$ to $I_{C_{MM_{L'}}}$).

3. Determination of the information loss and the information deficit for each mapping.
4. Layout of the visual model M'.

Let the costs for creating the converter be K_c, the costs of the transformation itself K_t and the costs for the information loss during the conversion be K_i [5]. The overall costs of a transformation K_o calculates as $K_o = K_t + K_i + K_c/n$, where n is the number of models converted by the converter K.

The transformation process must not be completed in one single step. There are at least three possibilities to complete this operation (see figure 4 and [22]).

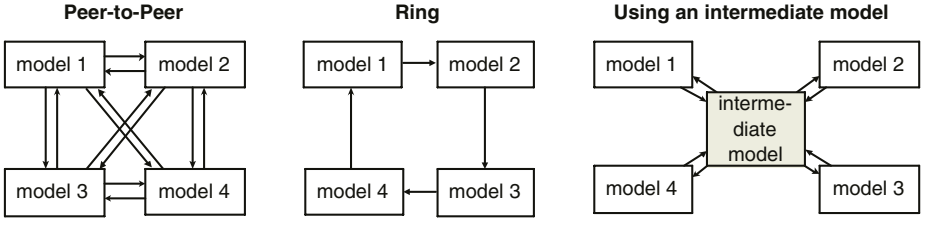


Fig. 4. Possibilities of converting a model (see [21], p. 5 figure 2.1; [22])

1. *Peer to peer migration*: This means, that models are translated directly from M to M' using a single converter for each pair of modelling languages. For this concrete mapping the transformation costs and costs of information loss are low. Since each converter works only for a pair of languages the costs of the converters are high (low n). Consequently, this strategy works for a small number of languages.
2. *Ring migration*: The model is transformed in a ring structure from M over $M_{i1}, M_{i2}, ..., M_{in}$ to M' in this case. The costs of this transformation are rather high, since up to $l - 1$ transformations are needed (where l is the number of languages). K_i is also high because the errors made during one transformation are added up. The costs of the converter are rather low, because only two converters are needed for each modelling language and it can be reused very often. The ring transformation can be used for a medium amount of languages.
3. *Migration with an intermediate model*: With this strategy a model is translated into an intermediate language L_i first and second transformed to the language L'. This involves two transformation steps. Thus, the transformation costs are higher than in the peer to peer approach while the information

[5] The information loss, which stems from the different semantics of the corresponding concepts of the two modelling languages is not important for this cost aspect since it is inherent to the migration process. Relevant for the transformation costs is the information loss that occurs because of the nature of the transformation process itself.

loss is low. Only l converters are needed for l languages so that the converters can often be reused (high n). The creation of the intermediate modelling language L_i is expensive, since it must cover all concepts of the modelling languages involved. So K_c/n is moderate. This approach is suited for a high number of modelling languages.

Generally, we follow the peer to peer migration. This allows on the one hand low costs for the model migration, but on the other hand requires a high number of converters. To reduce the number of converters, we use an intermediate metamodel, which provides a mapping between the concepts C_{MM_L} and $C_{MM_{L'}}$. With this mapping the model can be transformed directly from L to L' (see figure 5 for the unfolded process).

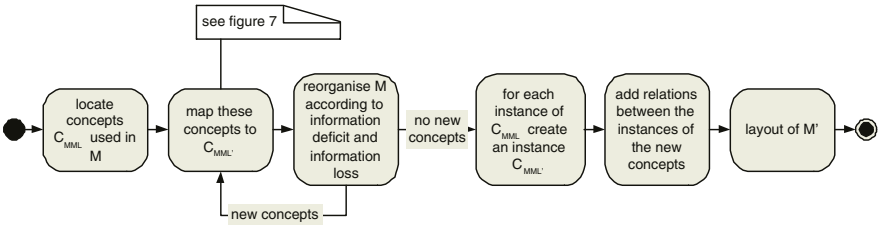


Fig. 5. Process model for the transformation process

The process starts with the determination of all meta model concepts C_{MM_L} in M. Second, these concepts can be mapped to the language L' (1:n mapping). This step reveals the information loss and deficit of this mapping. Third, because of this information deficit or loss the migration expert might decide to remodel M. Fourth, according to the mapping of the meta model concepts all instances of these concepts $I_{C_{MM_L}}$ in the model M are transformed to instances $I_{C_{MM_{L'}}}$ of the language L'. For each instance $I_{C_{MM_L}}$ the migration expert chooses the appropriate concept $C_{MM_{L'}}$ and creates an instance $I_{C_{MM_{L'}}}$. At this point of the transformation the information deficit and loss is known for the entire model. Fifth, the relations between the instances $I_{C_{MM_{L'}}}$ can be added. Lastly, M' can be laid out graphically.

As stated above the process cannot be completed automatically and always involves a human to address the pragmatic needs. Only the mapping of the language concepts C_{MM_L} and $C_{MM_{L'}}$ as well as the creation of the instances of $C_{MM_{L'}}$ can be automated. The migration expert must especially handle the information loss and must gain additional information to satisfy the information deficit. Additionally, the layout of the resulting diagram must be done manually[6].

[6] Automatic layout of diagrams is a mathematically difficult task. The main problem of laying out modelling graphs is to respect the specific graphical constraints defined in the modelling languages. Furthermore, modelling graphs contain a high number of vertexes, which make the layout process mathematically expensive (see [23]).

In the next section we present a meta model based migration approach including an intermediate meta model used to fulfil the step of mapping the language concepts to each other as well as to determine the information deficit and loss.

3 The Meta-model Based Migration Approach

3.1 The Intermediate Meta-model

The intermediate model is constructed for the purpose of the migration of models. Consequently, the concepts of different modelling languages that are viewed as semantically equivalent (for this purpose) must be related to each other.

For the sake of brevity we provide the intermediate model only for the modelling languages ERM (see [10]) and for simple UML class diagrams (see [12], p. 3-34ff). We use a UML class diagram for the description of the meta model itself.

The UML notation is extended by a special inheritance concept with the stereotype $<< eInheritance >>$. The graphical representation is a closed, filled arrow. All subclasses connected with this relation have the same meaning and are viewed as semantically equivalent and thus can be related to each other without information loss or deficit.

For the mapping of the concepts of the modelling languages the intermediate meta model is divided into two parts. First, the core meta model describes the core concepts needed to associate concepts of different modelling languages to each other. The core part is, second, extended by the concepts of the modelling languages involved using the $<< eInheritance >>$ relation. A stereotype with the name of the modelling technique is assigned to these concepts. Figure 6 shows the intermediate meta model.

We use the inheritance hierarchy to distinguish between general and specific concepts. The information deficit and loss can be determined by two factors. First, it is defined by additional attributes of the subclasses and second, by additional association and their specific counterparts aggregation and composition. This specification is summarised using the UML inheritance discriminator (see [12], p. 2-39).

The intermediate core meta model is the first part of the converter that maps the concepts of the modelling languages. The second part of the converter is the assignment of concrete language concepts to this core model. Third, we must describe how the actual mapping is derived from this meta model and how the information loss and deficit can be determined. These issues are discussed in the next subsection.

3.2 Process Model

After the concepts C_{MM_L} in M are determined (see figure 5) they can be located in the intermediate model by the algorithm depicted in figure 7.

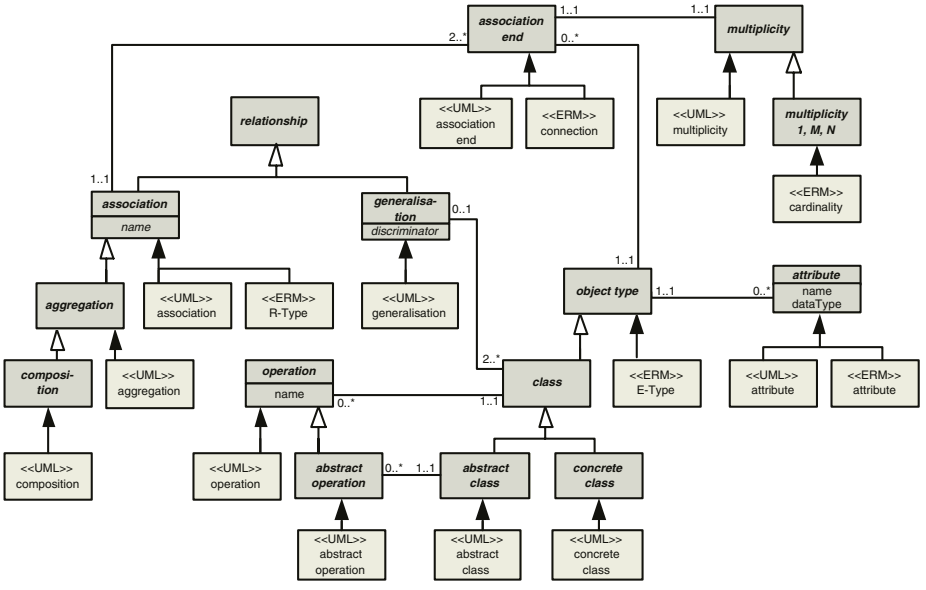


Fig. 6. The intermediate meta model

With this approach the destination concepts with no information loss and deficit result first. After this all special concepts (information deficit) are determined. If there is no result until this step, general concepts are found (information loss) and the whole process is repeated for these general concepts (information loss and deficit).

To illustrate the application of the complete framework an example is provided in the next subsection.

3.3 Application of the Meta-model Approach

Figure 8(a) shows the initial model, which is to be migrated to an ERM schema.

The first analysis step is to determine the UML concepts used in the initial model and to map these concepts to ERM concepts using the intermediate model. Table 1 shows the results of this step including the analysis of the information loss and deficit.

Because of this mapping the migration expert might decide to resolve the inheritance relation between `BusinessPartner`, `Customer` and `Supplier` (see figure 8(b)).

In this example there is already a 1:1 mapping between the UML and ERM concepts. So, the next step in the process model can be omitted and all instances $I_{C_{MM_{L'}}}$ can be created. After this step the migration expert lays out the model M' presented in figure 9. Finally, the user or creator of M verifies the resulting model.

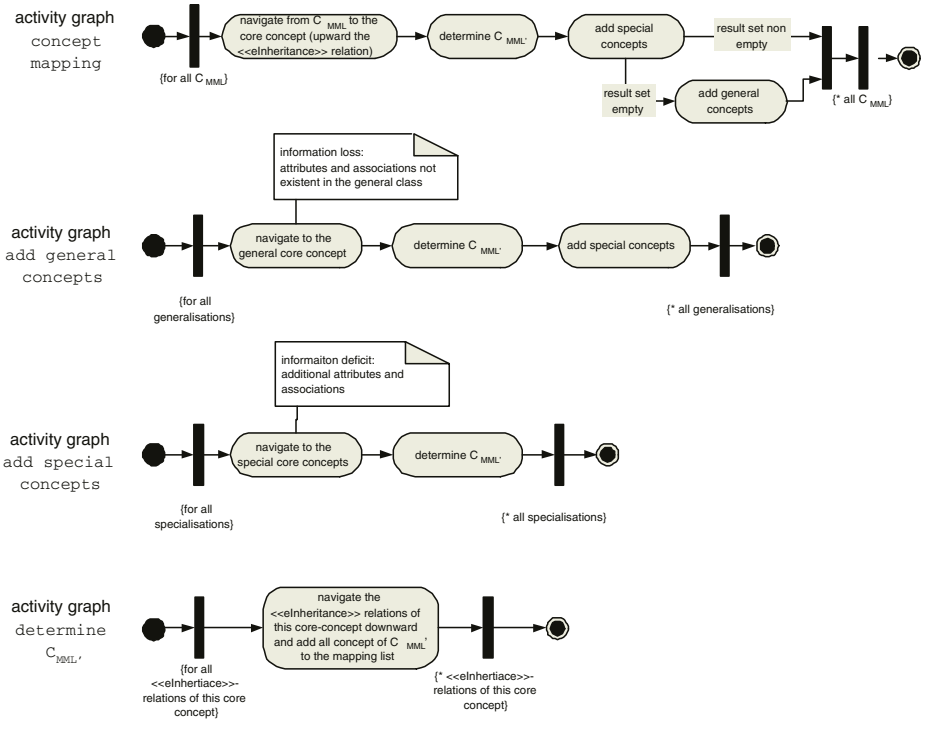
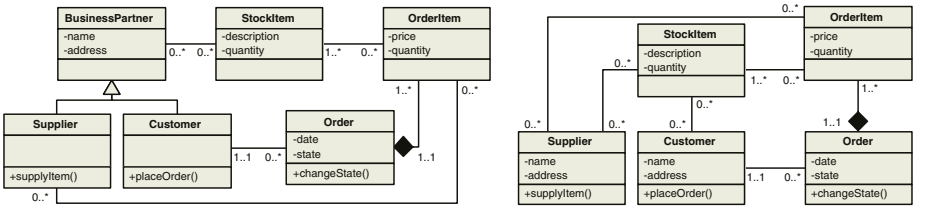


Fig. 7. Process to map the concepts of the language L to L' using the intermediate meta model



(a) Initial model (b) Remodelled

Fig. 8. Model to be migrated

4 Summary

We have shown in this paper that the migration of models of a material problem domain needs to address the syntactic, semantic and pragmatic aspects of these models. Furthermore, we have provided a framework for this purpose. It consists of a role model, a process model and an – interchangeable – meta model based migration approach.

Table 1. Mapping of the UML concepts to ERM concepts

UML concept	ERM concept	information loss	information deficit
`concrete class`	`E-type`	operations	none
`abstract class`	`E-type`	abstract operations; operations	none
`operation`	-	complete	none
`attribute`	`attribute`	none	none
`generalisation`	`association`	relation to class	relation to association end, name
`composition`	`association`	dependency	none
`multiplicity`	`cardinality`	none	type of cardinality

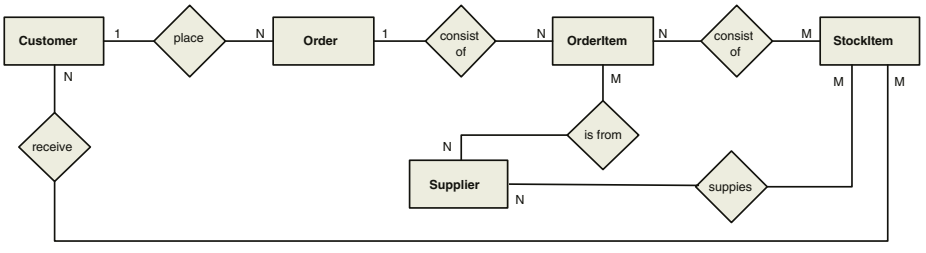


Fig. 9. Resulting Model

The role model of the migration process describes at least four roles, the modeller, the user (of the initial model M), the migration expert and the end-user (of the resulting model M'). In a material problem domain a complete migration of the model with its full semantic and pragmatic meaning is only feasible iff the migration expert is either the modeller itself (e. g. during the system development process) or the user of M or if the later verifies the resulting model. In all other cases there is a loss of information.

The process model consists of three core activities. First, the migration expert analysis the model M forming an internal model of it. Second, this model is transformed to the destination language L' using the mapping of the meta model concepts C_{MM_L} and $C_{MM_{L'}}$. The migration expert handels the information loss and deficit during this phase. Finally, the resulting model M' is verified by the creator or the user of M. Only this chain of activities ensures that the resulting model contains the same semantic and pragmatic information as the initial model M.

In this study the model migration framework was examined focusing on the aspect of modelling languages. Further research must concentrate on the aspect of modelling languages including the scientific findings of the semiotics. Additionally, we must incorporate the meta model of the modelling process into the framework, because it may lead to different models. Therefore, it may has an impact on the model migration.

References

1. Hofstede, A., Verhoef, T.: On the feasibility of situational method engineering. Information Systems **22** (1997) 401–422
2. Nuseibeh, B. A.: A multi-perspective framework for method integration. PhD thesis, University of London, Department of Computing (1994)
3. Harmsen, A. F.: Situational method engineering. Master's thesis, University of Twente (1997)
4. Brinkkemper, S., Saeki, M., Harmsen, F.: Assembly techniques for method engineering. Number 1413 in Lecture Notes of Computer Science, Springer (1998) 381–400
5. OMG: MDA guide version 1.0.1. Document Number: omg/2003-06-01, `http://www.omg.org/docs/omg/03-06-01.pdf`, Download: 19.02.2003 (2003)
6. Sprinkle, J.M.: Metamodel driven model migration. PhD thesis, Graduate School of Vanderbilt University, Nashville, Tennessee (2003)
7. Czarnecki, K., Helsen, S.: Classification of model transformation approaches. In: OOPSLA 2003, 2nd OOPSLA Workshop on Generative Techniques in the context of the Model Driven Architecture `http://www.softmetaware.com/oopsla2003/czarnecki.pdf`, Download: 27.02.2004 (2003)
8. Stachowiak, H.: General model theory. Springer, Vienna (1973) – in German
9. Dresbach, S.: Modeling by construction: A new methodology for constructing models for decision support. `http://www.winfors.uni-koeln.de/pub/wp_mbc.htm`, Download: 25.11.2003 Working Paper 5, Chair for Business, Information Science and Operations Research, University of Cologne, Germany (1995)
10. Chen, P. P. S.: The entity relationship model – toward a unified view of data. ACM Transactions on Database Systems **1** (1976)
11. Keller, G., Nüttgens, M., Scheer, A. W.: Semantic process modelling on the basis of event-driven process chains (EPC). `http://www.iwi.uni-sb.de/Download/iwihefte/heft89.pdf`, Download: 27.02.2004, Technical Report 89, Publications of the Institute of business and computing, University of Saarland (1992) – In German
12. OMG: OMG unified modeling language specification: Version 1.5. Document Number: formal/03-03-01, `http://www.omg.org/cgi-bin/apps/doc?formal/03-03-01.pdf`, Download: 27.02.2004 (2003)
13. Guizzardi, G., Pires, L. F., van Sinderen, M. J.: On the role of domain ontologies in the design of domain-specific visual modeling languages. `http://www.cis.uab.edu/info/OOPSLA-DSVL2/Papers/Guizzardi.pdf`, Download: 06.11.2003 In: OOPSLA 2002 Second Workshop on Domain Specific Visual Languages, November 4, 2002, Seattle, WA, USA (2002)
14. Harel, D., Rumpe, B.: Modeling languages: Syntax, semantics and all that stuff, part I: The basic stuff. `http://wisdomarchive.wisdom.weizmann.ac.il/archive/00000071/01/00-16.ps`, Download: 22.11.2003 Technical report msc00-16, Weizmann Institute Of Science (2000)
15. Schefe, P.: Software engineering and epistemology. Informatik Spektrum **22** (1999) 122–135 – In German
16. Greiffenberg, S.: Methods as theories in the business and informations science discipline. In: Proceedings of the 6th international conference of business and information science, Physica (2003) 947–967 – In German

17. Gurr, C.A.: Effective diagrammatic communication: Syntactic, semantic and pragmatic issues. Journal of Visual Languages and Computing **10** (1999) 317–342
18. Becker, J., Rosemann, M., von Uthmann, C.: Guidelines of business process modeling. Volume 1806 of Lecture Notes in Computer Science, Springer (2000) 30–49
19. Weber, R.: Ontological foundations of information systems. Coopers & Lybrand (1997)
20. Schütte, R.: Guidelines of reference modelling: construction of configurable and adaptable models. Gabler (1998) – In German
21. Su, S.Y.W., Fang, S.C., Lam, H.: An object-oriented rule-based approach to data model and schema translation.
`http://citeseer.nj.nec.com/ su93objectoriented.html`,
Download: 10.10.1993, Technical Report TR-92-015, Database Systems Research and Development Center, University of Florida, USA (1992)
22. Wüstner, E., Hotzel, T., Buxmann, P.: Converting business documents: A classification of problems and solutions using xml/xslt. In: Proceedings of the 4th International Workshop on Advanced Issues of E-Commerce and Web-based Systems (WECWIS2002) (2002)
23. Jünger, M., Mutzel, P.: Automatic layout of diagrams. OR News (2001) 5–12 – In German