5-2-2012

ASPECTS OF THE CLASSIFICATION DEPENDENCY IN THE INTEGRATION OF STRUCTURAL KARLSTAD ENTERPRISE MODELLING SCHEMATA

Peter Bellström
Karlstad University

ASPECTS OF THE CLASSIFICATION DEPENDENCY IN THE INTEGRATION OF STRUCTURAL KARLSTAD ENTERPRISE MODELLING SCHEMATA

Bellström, Peter, Karlstad University, Universitetsgatan 2, 651 88 Karlstad, Sweden,
Peter.Bellstrom@kau.se

Abstract

In this paper, we address the classification dependency used to illustrate that one concept is an instance-of another concept. In doing so, we present four important aspects of the classification dependency in the integration of implementation-neutral Karlstad Enterprise Modelling schemata. First and second, the classification dependency can be used not only to recognise and resolve power types but also to recognise and resolve homonyms. Third, in inference rules, the classification dependency can be used to deduce both new concepts and dependencies from already existing ones. Fourth, the classification dependency can be used to counter the occurrence of semantic loss and maintain the vocabulary used in the source schemata. The classification dependency can even be used to semantically enrich the integrated conceptual schema. The four aspects of the classification dependency should also be viewed as important aspects to take into account during the development of a semi-automatic method for the integration of implementation-neutral structural EM schemata. Finally, by applying the classification dependency, several problems, such as power types and homonyms, might be recognised that otherwise could pass unnoticed in the integration process.

Keywords: Schema Integration, Classification dependency, Karlstad Enterprise Modelling Approach, Power Types, Homonyms, Inference Rules, Semantic Loss, Implementation-Neutral Schemata.

1 Introduction

When modeling an information system, the requirements are often illustrated in a set of conceptual schemata that are gathered from various information sources. These schemata can be divided into structural and behavioural schemata. The structural schemata illustrate both what should be stored in the future database and what data the future information system needs to provide its services. In contrast, the behavioural schemata illustrate what services the future information system is going to provide. Dealing with heterogeneous system requirements often results in schemata describing the future information system on different levels of abstraction (schema level and instance level). For instance, in one schema, one concept might be illustrated as a specialisation of another concept while the same concept might in the following schema be illustrated as an instance-of another concept. These types of problems need to be addressed and resolved when modelling an information system.

In this paper, we focus on the classification dependency in the integration of implementation-neutral structural Karlstad Enterprise (EM) modelling schemata. Schema integration is a research topic that has engaged and occupied researchers for several decades. Some researchers (e.g. Rahm and Bernstein, 2001) claim that it started in the early 1980's, while others claim that it started a few years earlier in the late 1970's, (e.g. Bellström, 2005). Despite this long tradition of research, there are still several research questions that are left unanswered. One such question deals with multi-level abstractions and the classification dependency in the integration of structural EM schemata. The multi-level problem is highlighted in Doan et al. (2004), where the authors in relation to semantic integration describe the schema heterogeneity (discrepancy) problem as follows: "schema elements of one source can be represented as data in another." (p. 11), and in He and Ling (2004), the authors describe the schema discrepancy (heterogeneity) problem as "the same information is modeled as data in one database, but metadata in another." (p. 245). Schema heterogeneity is also a problem that still occasionally arises in practice (Krishnamurthy, 1991; Miller, 1998). In He and Ling (2004), the authors even claim that schema discrepancy is something that arises frequently and not just occasionally.

One of the most quoted and referenced definitions of 'schema integration' is given in Batini et al. (1986), where the authors state that 'schema integration' is "the activity of integrating the schemas of existing or proposed databases into a global, unified schema." (p. 323). In a traditional schema integration process, the source schemata are processed in at least four steps (Batini et al., 1986): pre-integration (the source schemata are prepared for integration), comparison of the schemata (the source schemata are compared, aiming to recognise similarities and differences), conforming the schemata (the source schemata are adjusted to resolve the recognised similarities and differences) and finally merging and restructuring (the source schemata are superimposed into one schema and restructured to avoid redundancy). In the literature, the second phase, comparison of the schemata, is often pointed out as being very difficult (Doan et al., 2004; Ekenberg and Johannesson, 1996; Lee and Ling, 2003) and important (Song, 1995), while the third phase, conforming the schemata, as critical (Lee and Ling, 2003) and a key issue (Spaccapietra and Parent, 1994) in the integration process. In this paper, we also address the classification dependency within these two phases.

In this paper, we present four aspects of the classification dependency in the integration of implementation-neutral structural EM schemata. Within each aspect, we address issues that have to be dealt with while doing multi-level schema integration (schema level and instance level). The four aspects addressed in this paper are: *power types*, *homonyms*, *inference rules* and *semantic loss*. The four aspects should also be viewed as important aspects to take into account during the development of a semi-automatic method for integration of implementation-neutral structural EM schemata.

This paper is structured as follows: section two presents the EM approach and puts focus on the structural aspects (static dependencies). In section three, we present the main contributions of this paper: the four aspects of the classification dependency. In section three we also address related work and distinguish it from our own. Finally, the paper closes with a summary and conclusions.

2 The Karlstad Enterprise Modelling Approach

The Karlstad Enterprise (EM) Modelling approach refers to a modelling approach developed at Karlstad University, Sweden. By applying EM, the designer and domain expert can model and design three aspects of an information system using one and the same modelling language. These aspects are: the *pragmatic*, the *semantic* and the *syntactic* aspects (dependencies and elements).

The pragmatic dependencies (pragmatic aspects) are used to model goals, problems and opportunities together with both negative and positive influences (Gustas and Gustiené, 2008).

The semantic dependencies (semantic aspects) are used to model both the static aspects (see Figure 1) and the dynamic aspects of an information system and its domain. In this paper, we will mainly focus on the static classification dependency used to illustrate that one concept is an 'instance-of' another concept. Additionally, the dynamic aspects are divided into communication dependencies, illustrating actors, their actions and communication flows, and state dependencies, illustrating states and conditions between actions (Gustas and Gustiené, 2004).

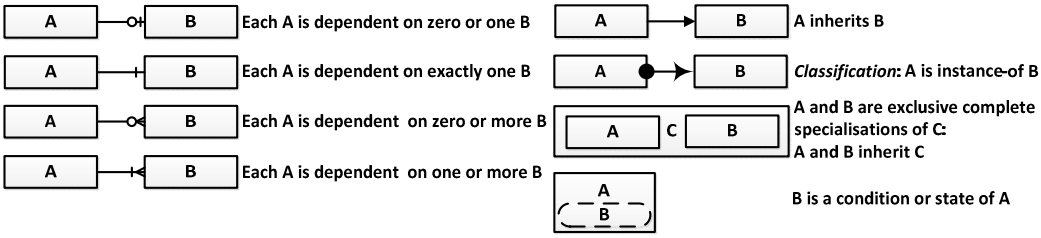

Figure 1. Static dependencies, structural aspects, in EM (adapted and modified from Gustas and Gustiené, 2004)

The syntactic aspects are a set of CASE-tool dependent syntactic elements where new elements can be added on demand. In other words, a set of syntactic elements used in one project is not necessarily the same set of syntactic elements used in another project: the list is not exhaustive (Gustas and Gustiené, 2004).

Several descriptions and definitions of EM have been given. For instance in Gustas and Gustiené (2004), the authors describe EM as a generalisation and extension of system analysis and design, while in Vernadat (1996), the author instead describes EM as an approach that deals with modeling and integration of business processes.

In Figure 1, the static dependencies (structural aspects) of EM used in this paper are illustrated. It should be noted that in EM, the only primitives that are given a name (a label) among the structural aspects are concepts, which are drawn as boxes. This should be compared with more traditional modeling languages for conceptual database design such as the Entity-Relationship modelling language (ER) (Chen, 1976) in which entities, relationships and attributes all are given names (labels). In EM, dependencies (connections/links) between two concepts are drawn as lines. At the beginning and the end of each dependency, additional symbols might also be added to illustrate the cardinality and/or a specific dependency.

In EM, one does not distinguish between entities (classes) and attributes (properties) but instead puts focus on concepts and dependencies between concepts. At the same time, this indicates that in this approach, we focus on content (what) rather than implementation issues (how), meaning the schemata are implementation-neutral. Finally, in EM, a concept might be interpreted in different ways depending on the dependency in focus.

3 Aspects of the Classification Dependency

As indicated in the introduction, integrating schemata that include not only modelling elements (model level) but also data (instance level) is one of the research directions that until recently has been neglected in favour of schema integration on the model level. Some initiatives with the Unified Modeling Language (UML) (e.g. Gonzales-Perez and Henderson-Sellers, 2006), the Web Ontology Language (OWL2) (e.g. Neumayr and Schrefl, 2009), and the Entity-Relationship (ER) modelling language (e.g. He and Ling, 2004; Neumayr and Schrefl, 2011) have been conducted. However, the UML, the ER and OWL2 are all techniques used in relation to the implementation *dependent* level, while EM is instead used to model and illustrate the implementation *independent* level. Other issues that are worth mentioning are that the UML has 13 diagram types (OMG, 2012) that have to be checked for consistency and integrity. ER has only one diagram type, but it can only be used to model the static aspects of an information system and it lacks the classification dependency. OWL2 (OWL, 2009) is simply an ontology language and therefore focuses on formally defined meaning. In the EM approach, we only have one schema type that could be used to illustrate both the static and the behaviour dependencies of an information system. Before addressing the four aspects of the classification dependency: *power types*, *homonyms*, *inference rules* and *semantic loss*, we quote Hoppenbrouwers et al. (2005) in which the authors state in relation to the creation of models that "*When discussing models with stakeholders and informants, in particular when trying to establish a common understanding, it is sensible to discuss different scenarios and alternatives to the model being considered. Doing so leads to an exploration of the meaning and impact of the model taking shape, and also leads to improved mutual understanding*" (p. 271).

The statement given in Hoppenbrouwers et al. (2005) justifies even more the use of the classification dependency, since by applying it, we are able to "*discuss different scenarios and alternatives to the model being considered*" (p. 271), which is one of the most important tasks in schema integration.

Finally, some words about the used method for comparison of the source schemata are needed. First of all, in the approach, we use binary integration (Batini et al., 1986; Batini et al., 1992) since there are always two source schemata that are compared, conformed and integrated. Secondly, we do not only compare concept names but also concept neighbourhoods (see Bellström, 2010a; 2010b; Bellström and Vöhringer, 2009; 2011) and in doing so, several differences and similarities between two source schemata might be recognised that otherwise could pass unnoticed.

3.1 Power Types

As mentioned in Neumayr and Schrefl (2011), the notion of power type was first introduced in Cardelli (1988), in which the author described a power type as "If A is a type, then Power(A) is the type whose elements are all the subtypes of A; if B has type Power(A) then B is a subtype of A" (p. 72). Power types have also been addressed and described in relation to the UML. In Martin and Odell (1998) for instance, the authors describe a power type as "a type whose instances are subtypes of another type" (p. 252) and in the official OMG document that describes the UML (OMG UML, 2010), a power type is described as "a class whose instances are subclasses of another class" (p. 78).

In a traditional conceptual database design modelling language such as the ER, power types are not described or explicitly stated within the schemata. If power types are even recognised, they might be described in natural language. This is a weakness because if power types are not addressed and illuminated, we might run into problems in the integration process or even end up with an incorrect integrated conceptual schema. In the long run, we might also end up processing and storing redundant data in the future implemented information system. Therefore classification dependencies should be illustrated in the schemata even though they could be experienced as inconvenient (Gustas, 2010). In the remainder of this section, the example schemata and integration scenario illustrated in Figure 2 are analysed and discussed.

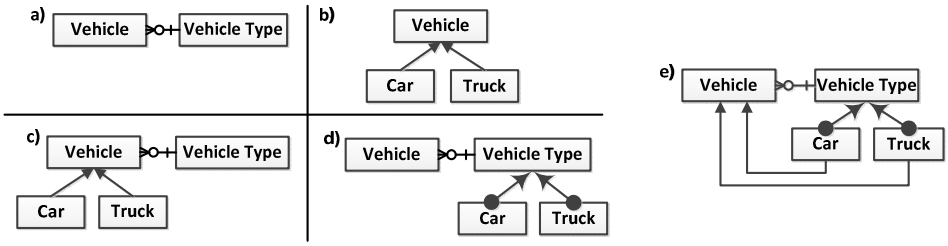

Figure 2. Recognition (a-d) and resolution (e) of power type

Conducting schema comparison on the schemata illustrated in Figure 2a and Figure 2b, we start by comparing each of the concept names within schema one (Figure 2a) and schema two (Figure 2b). In doing so, we find out that *Vehicle* in Figure 2a is equal to *Vehicle* in Figure 2b according to their names (*a.Vehicle = b.Vehicle*). Comparison of concept neighbourhoods does not result in any problems and therefore the two schemata are integrated (Figure 2c). In the following step we continue by comparing the concept names within Figure 2c and Figure 2d, which results in four matches *c.Vehicle = d.Vehicle*, *c.Vehicle Type = d.Vehicle Type*, *c.Car = d.Car* and *c.Truck = d.Truck*. However, in the following step, comparison of concept neighbourhoods, we find out that in schema one (Figure 2c) *Car* and *Truck* are both subtypes to *Vehicle*, while in schema two (Figure 2d) *Car* and *Truck* are both instead classified as *Vehicle Type*. In other words, a comparison of concept neighbourhoods yields *c.Car ≠ d.Car* and *c.Truck ≠ d.Truck*, which indicates a difference between the two source schemata. That being said, in the following step, analysing and comparing the results from both the comparison of concept names and the comparison of concept neighbourhood, a power type would instead be indicated. Looking back at the description of what a power type is, (OMG UML, 2010) describe it as "a class whose instances are subclasses of another class" (p. 78). In our example, this fits since *Car* and *Truck* are both classified as *Vehicle Type* and subclasses of *Vehicle*; in other words, *Vehicle Type* is a power type. Therefore the schemata illustrated in Figure 2c and Figure 2d can be integrated into the schema as illustrated in Figure 2e in which *Car* and *Truck* are modelled and illustrated as both classified as *Vehicle Type* and as specialisations of *Vehicle*.

Given the concept names and dependencies in Figure 2c and Figure 2d, the following rule for the recognition of power types in the integration of structural EM schemata might be stated:

If *schema one contains an atomic concept A, a composite concept AB (A followed by another word B with a 1:M cardinality to A), a specialisation of concept A called C (atomic)* ***and*** *schema two contains an atomic concept A, a composite concept AB (A followed by another word B with a 1:M cardinality to A), a concept classified as concept AB called C (atomic)* ***then*** *AB is most likely a power type.*

For a more detailed discussion about atomic and composite concept names, please see Bellström and Vöhringer (2009; 2011). In the rule, concept A might in the first source schema have several specialisations such as concept C, D and E (Figure 3a). However, to be sure that that AB is a power type C, D and E also have to be classified as concept A in the second source schema (Figure 3b).

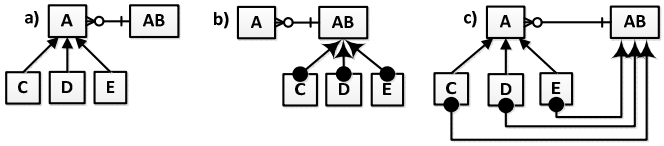

Figure 3. Illustration of rule for recognition of power types (a,b) and proposed method for resolution (c) of power types in the integration of structural EM schemata

3.2 Homonyms

Homonyms arise if the same concept name (label) is used for two or more concepts that represent different concepts in a given domain. In schema integration research, homonyms are often referred to

as name conflicts (Batini et al., 1986; Lee and Ling, 2003; Spaccapietra and Parent, 1994) and semantic conflicts (Song, 1995). Homonyms often arise when the vocabulary of terms is small (Batini et al., 1992) and when incomplete concept names are used (Kim and Seo, 1991).

Over the years, several approaches on how to recognise homonyms have been suggested, such as studying concept unlikeness (Batini and Lenzerini, 1984), comparing attribute values and domains (Larson et al., 1989), using semantic information (Bhargava and Beyer, 1992), using fuzzy thesauri (Mirbel, 1997), using algorithms (Palopoli, 2003), and comparing both concept names and concept neighbourhoods (Bellström and Vöhringer, 2009; 2011).

Several resolution methods for homonyms have also been suggested, such as renaming one or both concept names (Dupont, 1994; Mannino, 2007), prefixing concept names (Parent and Spaccapietra, 1998), introducing the dot-notation together with the inheritance dependency (Bellström, 2005; 2006a) and standardising names (Lawrence and Barker, 2001).

As indicated above, it is important to recognise homonyms and resolve them with a proper resolution method. This is due to the fact that if homonyms are not recognised, two different concepts that look equal on the surface, will most likely be merged into one concept resulting in an incorrect integrated schema since one concept is lost.

Nevertheless, in our research, we focus on how the classification dependency might be used in the recognition and resolution of homonyms, which is something that is overlooked in earlier research. This clearly differentiates this research from the above-mentioned methods and approaches.

In the remainder of this section, the example schemata and integration scenario illustrated in Figure 4 are analysed and discussed. Let us first assume that in the first iteration, schema one (Figure 4a) and schema two (Figure 4b) are compared resulting in two name matches: *a.Product = b.Product* and *a.Product Type = b.Product Type* and one neighbourhood match: the dependency with cardinality (0, *; 1, 1) is equal between the two matched concept pairs. Therefore the schema illustrated in Figure 4b is produced. In the iteration that follows, three name matches are recognised *b.Product = c.Product*, *b.Product Type = c.Product Type* and *b.TV = c.TV*. However, neighbourhood comparison does not result in any match but instead yields a difference: *b.TV* $\neq$ *c.TV* since in Figure 4b *TV* is both a specialisation of *Product* and classified as a *Product Type*, while in Figure 4c, *TV* is instead classified as a *Product*. In other words, the same concept name is used to represent two different concepts within the given domain. This means that homonyms have been recognised through the use of the classification dependency. In a traditional integration method, this would instead indicate a difference of the used dependencies resulting in inconsistent use of concept names in the integrated schema.

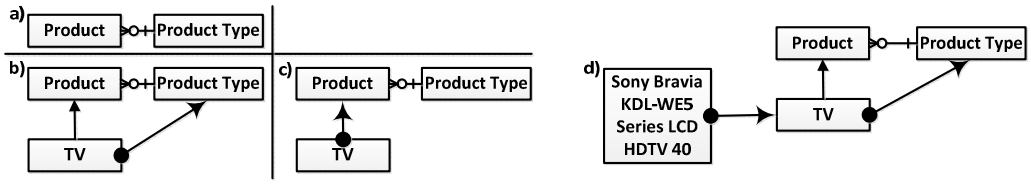

Figure 4. Recognition (a-c) and resolution (d) of homonyms

Analysing the schema in Figure 4b, we can conclude that *Product Type* is a power type since its instance (*TV*) is a subtype of another type (*Product*). We can therefore conclude that the concept names used in Figure 4b are most likely complete in the domain that the information system is being designed for. Having drawn that conclusion, we can also assume that the use of concept name *TV* in Figure 4c is incomplete and we therefore need to talk to the domain experts and specify a more precise concept name to use in the integrated schema. For simplicity, we assume that this was done and it was concluded that *TV* in Figure 4c should instead be named *Sony Bravia KDL-WE5 Series LCD HDTV 40* and that *Sony Bravia KDL-WE5 Series LCD HDTV 40* should be classified as a *TV* resulting in the integrated schema illustrated in 4d.

Given the concept names and dependencies in Figure 4b and Figure 4c, the following rule for the recognition of homonyms in the integration of structural EM schemata might be stated:

If *schema one contains an atomic concept A, a composite concept AB (A followed by another word B with a 1:M cardinality to A), a specialisation of concept A called C (atomic) and C is also classified as AB* ***and****, schema two contains an atomic concept A, a composite concept AB (A followed by another word B with a 1:M cardinality to A) and an atomic concept C classified as A* ***then*** *C in schema one and schema two are homonymic*.

This is a rather unique rule since, if a traditional integration method and/or modelling approach such as the ER or the UML with a traditional integration method had been used, the differences between the schemata illustrated in Figure 4b and Figure 4c would instead indicate a dependency conflict. Furthermore, the proposed resolution method would be to remove the classification dependency between *Product* and *TV* and use the schema in Figure 5b as the integrated schema, resulting in at least one problem: loss of a dependency.

It should be noted that the rule is also applicable even if one of the schemata does not explicitly illustrate a power type (the classification dependency between C and AB in Figure 5a is missing). However, if that is the case the strength of the rule is lost, which could lead to problems when resolving the differences between the source schemata.

It should also be noted that the rule indicates that two concepts on different levels (model and data) are recognised and we therefore need to introduce a more specific concept name that includes the original concept name (often with both a prior and a following addition e.g. prior *Sony Bravia KDL-WE5 Series LCD HF* (D) + original *TV* (C) + following *40* (E)).

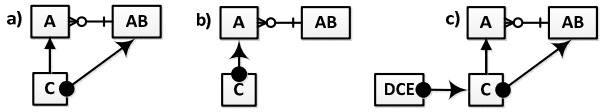

Figure 5. Illustration of rule for the recognition of homonyms (a,b) and proposed method for the resolution (c) of homonyms in the integration of structural EM schemata

3.3 Inference Rules

Inference rules are useful in the schema integration process since they open up the possibility to deduce new concepts and dependencies from already existing ones. Amongst other things, inference rules can be used in relation to weaker dependencies, the inheritance dependency, semantic quality improvement and the classification dependency (Bellström, 2006b). Inference rules are also applicable for both static aspects and dynamic aspects in the integration process (see Gustas, 2005). However, in this section we only consider the classification and the inheritance dependency within the static aspects.

One of the most useful inference rules within schema integration is described in formula 1 (see Bellström, 2006b; Gustas, 2010):

If A ●→ B and B ➡ C then A ●→ C **(1)**

Formula 1 should be interpreted as follows. If A is classified (●→) as B and B inherits (➡) C then A is classified as C. For instance if *ECIS 2012* is classified as *Conference Publication* and *Conference Publication* inherits *Publication* then *ECIS 2012* is classified as *Publication* (Figure 6).

Figure 6. Inference rule including classification and inheritance dependencies

For schema integration, this rule is useful in at least two scenarios: *First*, when discussing different schemata and solutions with the domain experts trying to recognise and resolve similarities and differences between two source schemata. This is because the domain experts' model illustrates their own specific part of the information system in the given domain. The consequence is that one or two concepts and/or dependencies are often missing within one source schema, which is natural due to the focus on a specific part. For instance, one domain expert might model *ECIS 2012* as a *Conference Publication* while another domain expert might instead leave out *ECIS 2012* and focus on and model *Publication* and *Conference Publication* as a specialisation of (inheritance) *Publication*. *Second*, when either discussing a specific or a more generic concept in a given inheritance hierarchy. This is due to the fact that sometimes the domain experts and designers need to focus on a specific concept in an inheritance hierarchy without changing the schema structure.

Two other useful inference rules are given in Gustas (2005):

If C ●→ A, B ➡ A and ~ (C ●→ B) then C ●→ ¬ B, ¬ B ➡ A **(2)**

If C ●→ A, B ➡ A and ~ (C ●→ ¬ B) then C ●→ B **(3)**

In formula 2 and formula 3, ~ represents the logical negation (Gustas, 2005), ¬ the operation of concept negation (Gustas, 2010), ●→ the classification dependency and ➡ the inheritance dependency.

As mentioned in Gustas (2005), both inference rules given in formula 2 and 3 are useful in schema evaluation and integration. In that context, formula 2 might be used to automatically generate a new concept as follows:

If *The European Conference of Information Systems is classified as a Book, Educational Book is a specialisation of Book,* ***and*** *The European Conference of Information Systems is Not Classified as Educational Book* ***then*** *The European Conference of Information Systems is classified as Not Educational Book* ***and*** *Not Educational Book is a specialisation of Book.*

Formula 2 might for some domain experts clarify a specific state or condition for a given concept (see Figure 1). Formula 3 might instead be used to automatically detect that a given concept should in fact be classified as a more specific concept within a given inheritance hierarchy as follows:

If *The European Conference of Information Systems is classified as a Book, Educational Book is a specialisation of Book,* ***and*** *Not The European Conference of Information Systems is classified as Not Educational Book* ***then*** *The European Conference of Information Systems is classified as an Educational Book.*

As shown in this section, inference rules that include the classification and the inheritance dependency might, in the integration of structural EM schemata, be used not only to automatically generate new concepts, but also to automatically deduce a new classification dependency. Inference rules can also be used while changing focus between different concepts and levels in an inheritance hierarchy without needing to change the schema structure.

3.4 Semantic loss

The fourth and last aspect of the classification dependency addressed in this paper is semantic loss. Semantic loss is slightly different compared to the other three aspects since it is a problem that has to be illuminated during the entire integration process. However, in this paper, we focus on semantic loss in comparing and conforming the schemata. In relation to the integration of structural EM schemata, semantic loss was first addressed in Bellström (2009) and later, by the same author, described as "*a problem that occurs if one or several concept(s), including their names, and/or dependenc(y/ies) describing the meaning of a concept are lost in schema integration*" (Bellström, 2010a, p. 13). Semantic loss is often treated as synonymic with information loss, "*a problem that occurs if one or several concepts and/or dependencies that make it possible to store a real-world fact in the future*

database are lost in schema integration" (Bellström, 2010a, p. 14) and concept name compression "a problem that occurs if one or several concept names are merged (compressed) into one concept name, for example when choosing one of the concept names to represent a concept when trying to resolve a synonym conflict" (Bellström, 2006a, p. 46). However, information loss is a problem that has to be dealt with when conducting implementation *dependent* modelling, while semantic loss and concept name compression are two problems that have to be dealt with when conducting implementation *independent* modelling. Regarding information loss, some research has been conducted in relation to the resolution of similarities (e.g. Lee and Ling, 2003; Spaccapietra and Parent, 1994) and derived information (e.g. Dey et al., 1999; Rauh and Stickel, 1993). Examples of work related to concept name compression are given in Bellström and Carlsson (2004; 2006).

To illustrate and discuss semantic loss in relation to the integration of structural EM schemata, the example schemata given in Figure 7-8 will be addressed.

Let us therefore first of all take a closer look at the problem of power types and how it can be resolved and explicitly stated in an EM schema without causing semantic loss.

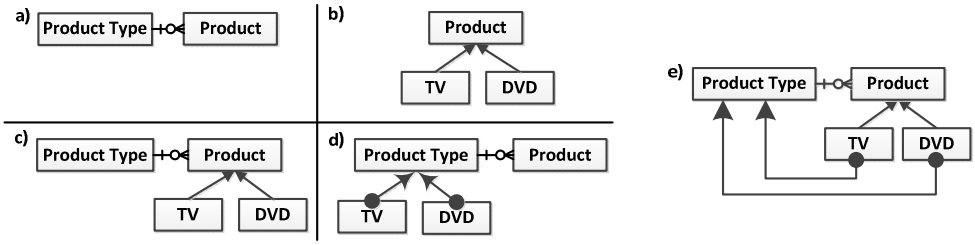

Figure 7. Semantic loss and power types

In Figure 7, four different approaches on how to model *Product* and *Product Type* are illustrated. However, although the semantics differ greatly between the four schemata, the possibility of storing the same data in the future database is still the same. This means that whatever the schemata illustrated in Figure 7, we are still able to store data about *Product* and *Product Type* in the future database. In this paper we do not focus on the implementation *dependent* level, the ability to store a fact, but on the implementation *independent* level and how the domain experts illustrate and model their specific part of an information system in a given domain. Looking at the schemata in Figure 7, we can therefore conclude that if the schema illustrated in Figure 7a is chosen, then we have semantic loss. This is the case since Figure 7a does not explicitly illustrate that *Product* might be specialised as *TV* and *DVD* or that *TV* and *DVD* might be classified as *Product Type*. Nevertheless, it is still possible to complement the schema with a natural language description stating that *Product Type {TV, DVD}*. But this might again be overseen in the integration process resulting in an incorrect schema and semantic loss. To maintain the vocabulary used in the source schemata, and at the same time to semantically enrich, instead of semantically impoverish, the global integrated schema, a combination of the source schemata illustrated in Figure 7c and Figure 7d should be used. In other words, the integrated schema should contain both the specialisation dependency and the classification dependency as illustrated in Figure 7e. In doing so, *TV* and *DVD* are both illustrated as specialisations of *Product* and are classified as *Product Type*.

Let us now take a closer look at the schemata illustrated in Figure 8. In Figure 8a, *DVD* is illustrated as both a specialisation of *Product* and classified as *Product Type*, whereas in Figure 8b, *DVD* is only classified as *Product*.

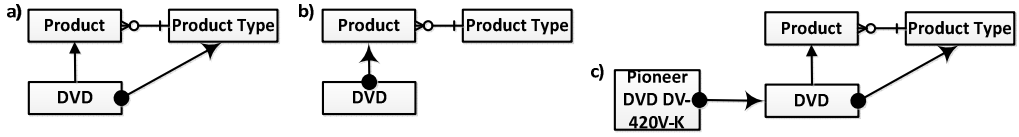

Figure 8. Semantic loss and homonyms

This means that if we were to integrate the schemata in Figure 8, *DVD* would be illustrated and modelled as both a specialisation of *Product* and at the same time classified as a *Product*, which clearly indicates that *DVD* is used homonymically. At the same time, this indicates that one level, the 'more precise' *DVD* instance-level, is missing. This means that a more precise concept name for *DVD* in Figure 8b has to be defined and an additional classification dependency has to be introduced between the concept with the new more precise name and the *DVD* concept.

Figure 8c illustrates the integrated schema in which the described homonyms are resolved introducing a more precise concept name *Pioneer DVD DV-420V-K* and a classification dependency between *Pioneer DVD DV-420V-K* and *DVD*.

As illustrated and discussed in this section, the classification dependency is a powerful and useful dependency in the integration of structural EM schemata. Not only can it be used to counter the occurrence of semantic loss but it can also be used to semantically enrich the integrated schema and maintain the vocabulary used in the source schemata.

4 Summary and Conclusion

In this paper, we have addressed four important aspects of the classification dependency in the integration of implementation-neutral structural EM schemata. In doing so, five contributions have been given to the research field.

First and second, we have shown how the classification dependency can be used to not only recognise and resolve *power types* but also to recognise and resolve *homonyms*. Third, we have shown how the classification dependency can, in *inference rules*, be used together with the inheritance dependency to deduce new concepts and dependencies. Fourth, we have shown how the classification dependency can be used to counter the occurrence of *semantic loss* and how to maintain the vocabulary used in the source schemata. Fifth, by applying the classification dependency, several problems, such as power types and homonyms, might be recognised that otherwise could pass unnoticed in the integration process.

Finally, the four aspects of the classification dependency should also be viewed as important aspects to take into account during the development of a semi-automatic method for integration of implementation-neutral structural EM schemata.

References

Batini, C. and Lenzerini, M. (1984). A Methodology for Data Schema Integration in the Entity-Relationship Model. IEEE Transactions on Software Engineering, 10 (6), 650-664.

Batini, C., Lenzerini, M. and Navathe, S.B. (1986). A Comparative Analysis of Methodologies for Database Schema Integration. ACM Computing Surveys, 18 (4), 323-364.

Batini, C. Ceri, S. and Navathe, S.B. (1992). Conceptual Database Design An Entity-Relatinship Approach. The Benjamin/Cummings Publishing Company, Redwood City.

Bellström, P. (2005). Using Enterprise Modeling for Identification and Resolution of Homonym Conflicts in View Integration. In Information Systems Development Advances in Theory, Practice and Education (Vasilecas O. et al. eds.), pp. 265-276, Springer Science, New York.

Bellström, P. (2006a). View Integration in Conceptual Database Design Problems, Approaches and Solutions. Licentiate Thesis. Karlstad University Press 2006:5.

Bellström, P. (2006b). Bridging the Gap between Comparison and Conforming the Views in View Integration. In Local Proceedings of the 10th East-European Conference on Advances in Databases and Information Systems (Manolopoulos, Y. et al. eds.), pp. 184-199, Publishing Centre Alexander Technological Educational Institute of Thessaloniki, Thessaloniki.

Bellström, P. (2009). On the Problem of Semantic Loss in View Integration. In Information Systems Development Challenges in Practice, Theory, and Education (In Barry, C. et al. eds.), pp. 963-974, Vol. 2, Springer Science, New York.

Bellström, P. (2010a). Schema Integration – How to Integrate Static and Dynamic Database Schemata. Dissertation. Karlstad University Press 2010:13.

Bellström, P. (2010b) A Rule-Based Approach for the Recognition of Similarities and Differences in the Integration of Structural Karlstad Enterprise Modeling Schemata. In Proceedings of the 3rd IFIP WG 8.1 Working Conference on The Practice of Enterprise Modeling (van Bommel, P. et al. eds.), pp. 177-189, Springer, Berlin.

Bellström, P. and Carlsson, C. (2004). Towards an Understanding of the Meaning and the Contents of a Database through Design and Reconstruction. In Proceedings of the 13th International Conference on Information Systems Development Advances in Theory, Practice and Education (Vasilecas, O. et al. eds.), pp. 283-293, Technika, Vilnius.

Bellström, P. and Carlsson, S. (2006). Language Aspects of Conceptual Database Design. In Proceedings of the 4th International Conference on Action in Language, Organisations and Information Systems (Lind, M., et al. eds.), pp. 211-223, Responstryck, Borås.

Bellström, P. and Vöhringer, J. (2009). Towards the Automation of Modeling Language Independent Schema Integration. In Proceedings of the International Conference on Information, Process, and Knowledge Management (Kusiak, A. and Lee, S. eds.), pp. 110-115.

Bellström, P. and Vöhringer, J. (2011). A Three-Tier Matching Strategy for Predesign Schema Elements. In Proceedings of The Third International Conference on Information, Process, and Knowledge Management (Jerman-Blazic, B. and Burdescu, D.D. eds.), pp. 24-29.

Bhargava, H.K. and Beyer, R. M. (1992). Automatic Detection of Naming Conflicts in Schema Integration: Experiments with Quiddities. In Proceedings of the Twenty-Fifth Hawaii International Conference on System Sciences (Shriver, B.D. ed.), pp. 300-310.

Cardelli, L. (1988). Structural Subtyping and the Notion of Power Type. In Proceedings of the Fifteenth Annual ACM SIGACT-SIGPLAN Symposium on Principles of programming languages, pp. 70-79.

Chen, P. (1976). The Entity-Relationship Model – Toward a Unified View of Data. ACM Transactions on Database Systems, 1 (1), 9-36.

Dey, D., Storey, V.C. and Barron, T.M. (1999). Improving Database Design through the Analysis of Relationships. ACM Transactions on Database Systems, 24 (4), 453-486.

Doan, A., Noy, F.N. and Halevy, A.Y. (2004). Introduction to the Special Issue on Semantic Integration. SIGMOD Record, 33 (4), 11-13.

Dupont, Y. (1994). Resolving Fragmentation Conflicts in Schema Integration. In Entity-Relationship Approach Business Modelling and Re-Engineering (Loucopoulos, P. ed.), pp. 513-532, Springer, Berlin.

Ekenberg, L. and Johannesson, P. (1996). A Formal Basis for Dynamic Schema Integration. In Conceptual Modeling – ER'96 (Thalheim, B. ed.), pp. 211-226, Springer, Berlin.

Gonzales-Perez, C. and Henderson-Sellers, B. (2006). A Powertype-Based Metamodelling Framework. Software & System Modeling, 5 (1), 72-90.

Gustas, R. (2005) Inference Rules of Semantic Dependencies in the Enterprise Modelling. In New Methods in Software Methodologies, Tools and Techniques (Fujita, H. and Mejri, M. eds.), pp. 235-251, IOS Press, Amsterdam.

Gustas, R. (2010). A Look Behind Conceptual Modeling Constructs in Information System Analysis and Design. International Journal of Information System Modeling and Design, 1 (1), 79-108.

Gustas, R. and Gustiené. P. (2004). Towards the Enterprise Engineering Approach for Information System Modelling Across Organisational and Technical Boundaries. In Enterprise Information Systems V (Camp, O. et al. eds.), pp. 204-215, Kluwer.

Gustas, R. and Gustiené. P. (2008). Pragmatic-Driven Approach for Service-Oriented Analysis and Design. In Information Systems Engineering: From Data Analysis to Process Networks (Johannesson, P. and Söderström, E. eds.), pp. 97-128, IGI Global.

He, Q. and Ling, T. W. (2004). Resolving Schematic Discrepancy in the Integration of Entity-Relationship Schemas. In Proceedings of ER 2004 (Atzeni, P. et al. eds.), pp. 245-258, Springer, Heidelberg.

Hoppenbrouwers, S.J.B.A., Proper, H.A. and van der Weide, Th.P. (2005). Understanding the Requirements on Modelling Techniqes. In: Advanced Information Systems Engineering (Pastor, O. and Falcão e Cunha, J. eds.), pp. 262-276, Springer, Berlin.

Kim, W. and Seo, J. (1991). Classifying Schematic and Data Heterogeneity in Multidatabase Systems. IEEE Computer, 24, 12-18.

Krishnamurthy, R., Litwin, W. and Kent, W. (1991). Language Features for Interoperability of Databases with Schematic Discrepancies. In Proceedings of the 1991 ACM SIGMOD international conference on Management of data, pp. 40-49.

Larson, J.A., Navathe, S.B. and Elmasri, R. (1989). A Theory of attribute equivalence in databases with application to schema integration. IEEE Transactions on Software Engineering, 15, 449-463.

Lawrence, R. and Barker, K. (2001). Integrating Relational Databases Using Standardized Dictionaries. In 16th ACM Symposium on Applied Computing, pp. 225-230. ACM Press.

Lee, M.L. and Ling, T.W. (2003). A Methodology for Structural Conflict Resolution in the Integration of Entity-Relationship Schemas. Knowledge and Information System, 5 (2), 225-247.

Mannino, M.V. (2007). Database Design, Application Development, & Administration. McGraw-Hill/Irwin, Boston.

Martin, J. and Odell, J.J. (1998). Object-Oriented Methods A Foundation. Prentice Hall, New Jersey.

Miller, R.J. (1998). Using Schematically Heterogeneous Structures. ACM SIGMOD Record, 27 (2), 189-200.

Mirbel, I. (1997). Semantic Integration of Conceptual Schemas. Data & Knowledge Engineering, 21, pp. 183-195.

Neumayr, B. and Schrefl, M. (2009). Multi-level Conceptual Modeling and OWL. In Advances in Conceptual Modeling - Challenging Perspectives ER 2009 Workshop (Heuser, G.A. and Pernul, G. eds.), pp. 189-199, Springer, Berlin.

Neumayr, B. and Schrefl, M., Thalheim, B. (2011). Modeling Techniques for Multi-level Abstraction. In The Evolution of Conceptual Modeling (Kaschek, R. and Delcambre, L. eds.), pp. 68-92, Springer, Berlin.

OMG UML. (2010). Object Management Group: OMG Unified Modeling Language (OMG UML), Superstructure. [Electronic], http://www.omg.org/spec/UML/2.3/Superstructure/PDF/ (21-03-2012).

OMG. (2012). Object Management Group: Introduction to OMG's Unified Modeling Language. [Electronic], http://www.omg.org/gettingstarted/what_is_uml.htm (21-03-2012).

OWL2. (2009). W3C OWL 2 Web Ontology Language Document Overview. [Electronic], http://www.w3.org/TR/owl2-overview/ (21-03-2012).

Palopoli, L., Terracina, G. and Ursino, D.: DIKE (2003). A System Supporting the Semi-automatic Construction of Cooperative Information Systems From Heterogeneous Databases. Software – Practice and Experience, 33, pp. 847-884.

Spaccapietra, S. and Parent, C. (1994). View Integration: a Step Forward in Solving Structural Conflicts. IEEE Transactions on Knowledge and Data Engineering, 6 (2), 258-274.

Rahm, E. and Bernstein, P.A. (2001). A Survey of Approaches to Automatic Schema Matching. The VLDB Journal, 10, 334-350.

Rauh, O. and Stickel, E. (1993). Searching for Composition in ER Schemes. In Entity-Relationship Approach - ER '93 (Elmasri, R.A. et al. eds.), pp. 74-84. Springer, Berlin.

Song, W. (1995). Schema Integration – Principles, Methods, and Applications. Dissertation. Stockholm: Department of Computer and Systems Sciences, Stockholm University.

Parent, C. and Spaccapietra, S. (1998). Issues and Approaches of Database Integration. Communications of the ACM, 41 (5es), 166-178.

Vernadat, F.B. (1996). Enterprise Modeling and Integration: principles and applications. Chapman & Hall.