On Relationships among Models, Meta Models and Ontologies

Motoshi Saeki† Haruhiko Kaiya‡
†Dept. of Computer Science, Tokyo Institute of Technology
Ookayama 2-12-1, Meguro-ku, Tokyo 152, Japan
saeki@se.cs.titech.ac.jp
‡Dept. of Computer Science, Shinshu University
Wakasato 4-17-1, Nagano 380-8553, Japan
kaiya@cs.shinshu-u.ac.jp

Abstract

In this position paper, we discuss the relationships among domain specific models, domain specific ontologies, meta models and a meta model ontology, in order to provide seamless semantics for both of models and meta models. By using the same semantic framework, we can detect semantic inconsistency included in models and meta models by using inference rules on the ontologies.

1 Introduction

Meta modeling techniques play an important role of developing model description languages suitable for problem domains, and they define abstract syntax of the model description languages. In fact, the UML meta model defines abstract syntax of all kinds of UML diagrams. However, meta models can express the logical syntactical structures of domain specific models (simply models, hereafter) only, and cannot specify the semantics of the models. There are many studies to combine modeling techniques with the formal methods having rigorous semantic basis, e.g. Class diagram and Z, and they provide transformation rules of a model description into a formal description like Z. In a broader sense, the transformation can be considered as an interpretation and the semantics basis of the formal description provide the formal semantics for the model description. However, these approaches are for general purpose and do not consider domain-specific properties. In particular, semantics where domain specific properties are embedded, i.e. domain-specific semantics is very significant for domain-specific modeling description languages.

On the other hand, there are a few works to provide the formal semantics for meta models, not for models. MEL (Method Engineering Language) [4] is a language to describe meta models and it gives the semantics to a meta model by using an ontology called method ontology. In [12], the schema of the rules to transform a model description such as a class diagram into a formal description like Z are defined with graph grammar and it presented that the semantics of a meta model can be considered as these transformation rules. However, in these works, none

of logical relationships to the semantics of the models as instances of the meta model could be found. The semantics of the meta model should be seamlessly connected to the semantics of the models.

In this position paper, we propose the usage of ontologies for domain specific semantics of models and for the semantics of meta models. By using two types of ontologies, we give semantics to both of a meta model and models as the instances of the meta model simultaneously. As a result, we can formally infer various properties of the meta model and the models in the same framework.

The rest of the paper is organized as follows. In the next section, we introduce the basic idea and clarify the relationships among models, domain-specific ontologies (domain ontologies), meta models and the ontology of meta models called meta model ontology. In our framework, a domain ontology play a role of a semantic domain for the models, while the meta model ontology provides a semantic basis on the meta models specifying abstract syntax of modeling description languages. To show the beneficial effects of our technique, we illustrate an example of consistency checking between a class diagram and a sequence diagram in the domain of Lift Control. By using this example, sections 3, 4 and 5 present the details of the semantic relationships between meta models and a meta model ontology, between models and domain ontologies, and between the meta model ontology and the domain ontologies, respectively. In section 6, we list up research agenda for future work.

2 Basic Idea

Ontology technologies are frequently applied to many problem domains nowadays [6, 14]. As mentioned in [11], we consider an ontology as a thesaurus of words and inference rules on it, where the words in the thesaurus represent concepts and the inference rules operate on the relationships on the words. Each concept of an ontology can be considered as a semantic atomic element that anyone can have the unique meaning in a problem domain [13]. That is to say, the thesaurus part of the ontology plays a role of a semantic domain in denotational semantics, and the inference rules help a model engineer (an engineer for developing models) in detecting lacks of model elements and semantically inconsistent parts during his or her model-development activities [8]. As mentioned above, we develop two types of ontologies; one is an ontology of meta models and another is an ontology of a problem domain, called domain ontology. A model is an instantiation of a meta model and it is semantically interpreted by the domain ontology. A domain ontology is an instantiation of the ontology of meta models. Figure 1 depicts the relationships among these ontologies, a model and a meta model. Models are instances of a meta model and their logical and syntactical structures should obey the meta model. For example, a sequence diagram shown in the left part of Figure 4 is an instance of the meta model of sequence diagrams shown in the left bottom part of Figure 3. Constraints appearing in the left part of the figure are imposed on the instances. For example, Constraints #2 attached to Meta Model are for the instances of the meta model, while Constraints #1 are for the instances of the model, i.e. M0 layer in MOF. Semantic mappings appearing in Figure 1, including a simple example, will be mentioned in section 2.

As mentioned in section 1, an ontology plays a role of a semantic domain in denotational semantics. Basically, our ontology is represented in a directed typed graph where a node and an arc represent a concept and a relationship (precisely, an instance of a relationship) between two concepts, respectively.

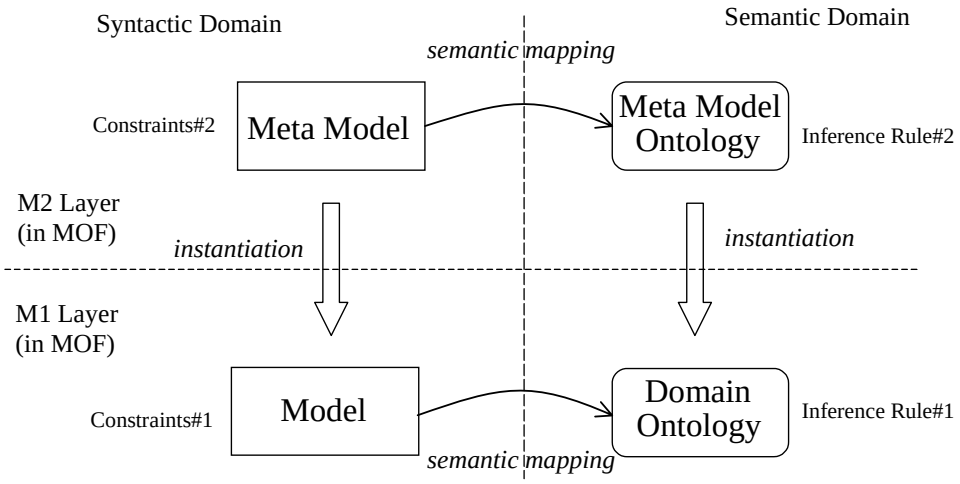


Figure 1: Relationships among Ontologies, a Model and a Meta Model

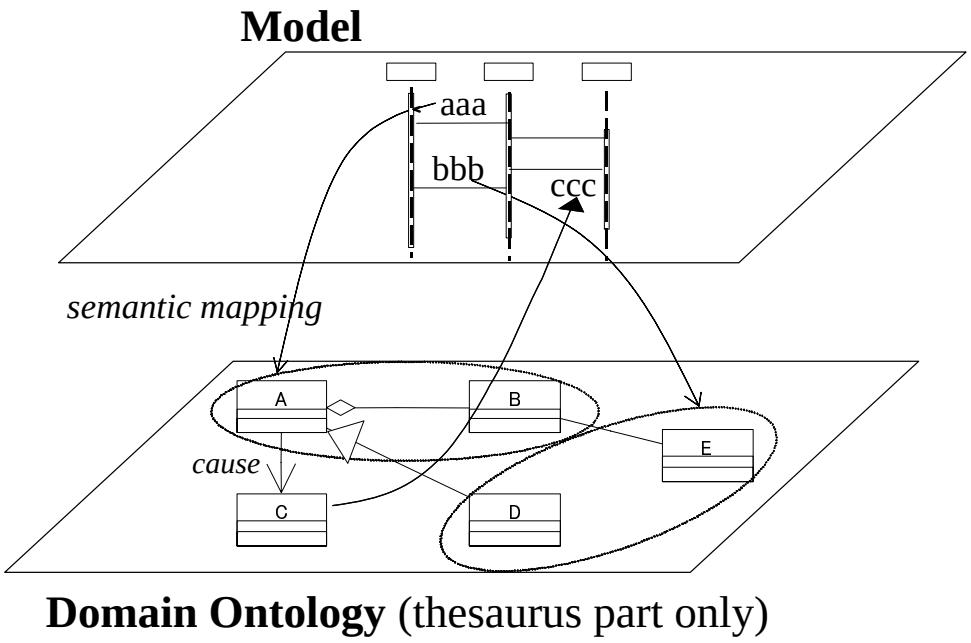


Figure 2: Mapping from a Model or a Meta Model to an Ontology

Below, let's consider how a modeling engineer uses a domain ontology for completing a model. During developing the model, the engineer should map its model element into atomic concepts of the ontology as shown in Figure 2. In the figure, the engineer develops a sequence diagram, while the domain ontology is written in the form of class diagrams. For example, the message "aaa" in the sequence diagram is mapped into the concepts A and B and an aggregation relationship between them. Formally, the engineer specifies a semantic mapping $\mathcal{F}$ where $\mathcal{F}$(aaa) = {A, B, an aggregation between A and B}. The sequence diagram may be incrementally improved, and logical inference on the ontology suggests to the engineer what part he or she should incrementally improve or refine. In the figure, although the model includes the concept A at the item "bbb", it does not have the concept C, which is caused by A. The inference resulted from "C is caused by A" and "A is included" suggests to the engineer that a model element having C, i.e. a message "ccc" should be added to the sequence diagram. In our technique, it is important what kind of relationship like "cause" should be included in a domain ontology for inference, and they result from an ontology of meta models as will be discussed in the next section.

3 Meta Models and Meta Model Ontology

The technique mentioned in the previous section can help a meta-model engineer (engineer for developing meta models) in constructing a meta model of semantically high quality. Suppose a simple example of a meta model consisting of simplified versions of Class Diagram and Sequence Diagram of UML. Following this meta model, another engineer, i.e. a model engineer constructs a class diagram of the information system to be developed, and then develops the sequence diagrams, each of which defines an scenario of the interactions among objects belonging to the classes appearing in the class diagram. The left part of Figure 3 illustrates the meta model written with Class Diagram. For simplicity, we use Class Diagram for specifying meta models. Although context-free aspects of abstract syntax of models can be defined with Class Diagram, some of context-sensitive aspects cannot. For example, the constraint that the same class name cannot appear more than once in a class diagram cannot be specified in Class Diagram by itself, and therefore we use first order predicate logic to express this kind of constraints. The above example constraint can be represented as follows;

$\forall c1, c2 \in Class \cdot ((name(c1) \neq name(c2)) \vee (c1 = c2))$

where $Class$ denotes a set of classes appearing in the class diagram and is from the meta model Class Diagram. This constraint is an example of Constraints #2 in Figure 1. Although the usage of OCL is natural to specify these kinds of constraints, we actually have used Prolog for implementation because of its query functions and interfaces to Java.

A meta model ontology can give the meaning of the elements of a meta model in the same way as the technique mentioned in the previous section. The right part of Figure 3 depicts a part of a meta model ontology, which is a simplified version of [7]. During developing a meta model, a meta-model engineer maps its elements into a part of the meta model ontology. In the example of the figure, the meta-model engineer maps Class and Object in the meta model into the concepts Class and Object respectively. Attribute is mapped into a set of {State, Data}, because attributes in a class play roles of the states of the object belonging to the class and of the local data that the objects have. Let $\mathcal{G}$ be the semantics mapping of the this example, and we can have

$\mathcal{G} : MetaModelElements \mapsto 2^{MetaModelOntologyElements}$

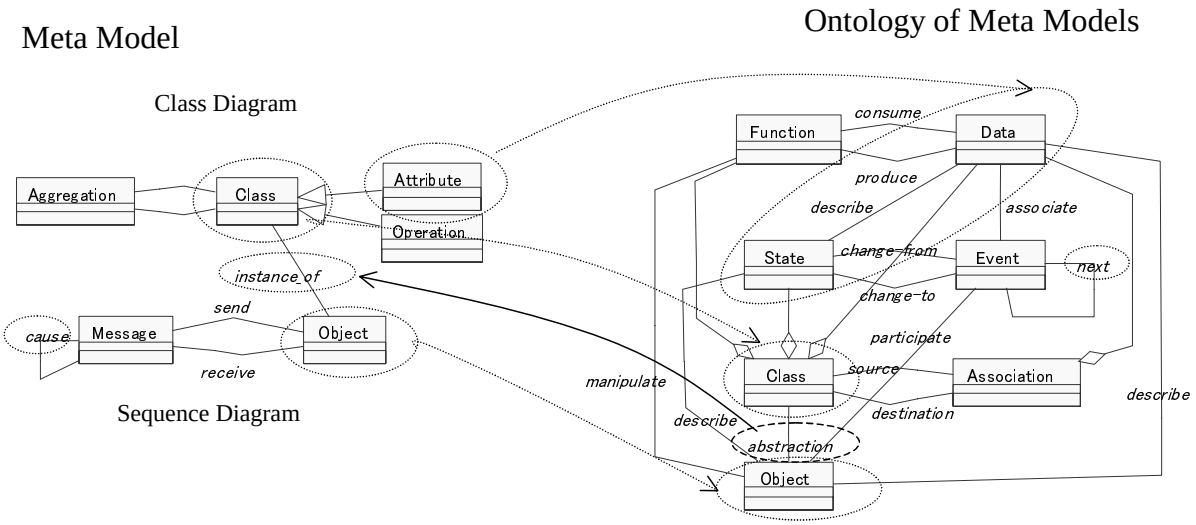


Figure 3: Meta Model and Meta Model Ontology

$\mathcal{G}$(Class) = Class, $\mathcal{G}$(Object) = Object, and $\mathcal{G}$(Attribute) = {State, Data},

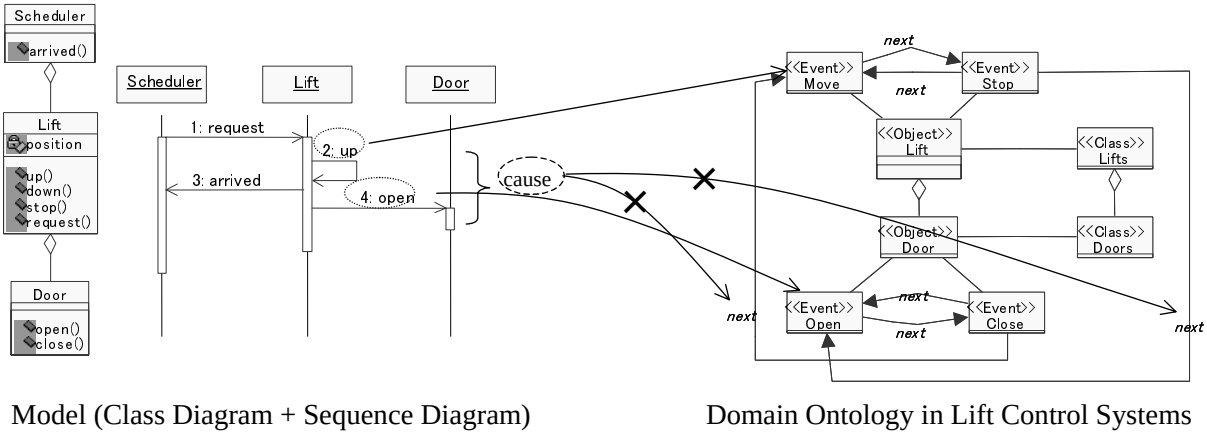
where $MetaModelElements$ and $MetaModelOntologyElements$ are a set of meta model elements such as Class and Message, and a set of elements of the meta model ontology such as Function and State, respectively.

Note that the example meta model is the result of assembling the meta models Class Diagram and Sequence Diagram by adding a new association *instance_of*. If a meta-model engineer did not add this new association, he or she got two isolated meta models as a result. This resulting meta model was not considered as a useful one, because a model engineer can construct class diagrams and sequence diagrams independently. The benefit of assembling Class Diagram and Sequence Diagram is the semantic relation between Class in Class Diagram and Object in Sequence Diagram to specify the behavior of instances of classes with sequence diagrams. The inference rule that a consistent meta model should not include isolated elements can be formalized with predicate logic and its detail was shown in [3]. This inference rule is an example of Inference Rule #2 in Figure 1.

The above example is a syntactical aspect of meta models. By establishing a semantic mapping meta model elements into the ontology, we can avoid producing semantically meaningless meta models [3]. The meta model ontology has several inference rules to keep semantic consistency on meta models. In this example, the inference rule "If both Class and Object concepts are included in a meta model, the association whose type is *abstraction* among them should be also included in the meta model" suggests to add a association that can be mapped to *abstraction*. See the figure 3. This mechanism is the same as the logical inference of on models that domain ontologies have, mentioned in the previous section.

4 Model and Domain Ontology

The left part of Figure 4 illustrates a model following the meta model of Figure 3, consisting of a class diagram and a sequence diagram. This example is a part of Lift Control System and the right part of the figure shows a part of a domain ontology of Lift Control System domain.



Model (Class Diagram + Sequence Diagram) Domain Ontology in Lift Control Systems

Figure 4: Model and Domain Ontology

We will explain stereotypes attached in classes of the domain ontology in the next section. The model engineer maps the messages “up” and “open” in the sequence diagram into Move and Open concepts of the ontology, during developing the model, as shown in Figure 4. And he or she tries to map *cause* relationship between the messages “up” and “open” into the association of type *next*. However, no events but Stop can be executed just after Move is executed because the domain ontology of Figure 4 specifies that Move has only one outgoing *next* relationship to Stop. Thus the inference rule on the ontology suggests that there are no *next* relationships between Move and Open and some events should be added to keep semantic consistency of execution order *next* relationship. Obviously, in this case the engineer should add the message Stop between “up” and “open”, which a Lift object sends to a Door object. The used inference rule is “If there is a *next* relationship between the concepts A and B in the domain ontology, then the elements mapped to A and B in the model, if any, should have the association mapped to *next* among themselves. This is also an example to detect semantic inconsistency, mentioned in section 2.

5 Meta Model Ontology and Domain Ontology

The meta model ontology can be considered as a meta model of domain ontologies. The stereotypes attached to the elements of a domain ontology express instantiated concepts of the meta model ontology. In the example of Figure 4, the element Move has the stereotype <<Event>> and it is an instantiation of Event in Figure 3. By providing the instantiation mechanism from the meta model ontology to domain ontologies, we can get the following two benefits.

1. The meta model ontology helps a domain engineer in developing domain ontologies, and it can play a role of a development methodology for ontologies like Object-Oriented Methodology for software development.

2. Although semantic processing may be possible in an extent, the efforts of establishing the mapping from elements to an ontology still remain human activities. Because only human can understand the meaning of a model and a meta model. However, some supports to develop the mapping are possible by using the combination of relations among the meta

model ontology, the domain ontology as its instance and the mapping from a meta model to the meta model ontology. In Figure 1, if all of the instantiation relations and semantic mappings are isomorphic, we can infer the semantic mapping from the model into the domain ontology, by using the other relations and other semantic mappings, i.e. the instantiation from the meta model to the model, the instantiation from the meta model ontology to the domain ontology and the semantic mapping from the meta model to the meta model ontology.

We will illustrate the second benefit, by using Figure 5. Suppose that a meta-model engineer produces a meta model combining Class Diagram and Sequence Diagram fragments and establishing the semantic mapping from *cause* association in the meta model into *next* relationship in the meta model ontology. The *next* relationship is instantiated to the "next" relationship between Stop and Open events in the domain ontology. In this situation, a model engineer develops a class diagram and a sequence diagram of Lift Control System following the meta model. He or she adds the messages "stop" and "open", which has the instance of *cause* association of the meta model, and then tries to map them into some elements of the domain ontology. Since the message concept in the meta model is semantically mapped into Message in the meta model ontology and Message is instantiated to Move, Stop, Open and Close in the domain ontology, the model engineer can select the suitable ontological elements out of these four elements. After the model engineer has mapped "stop" and "open" into Stop and Open respectively, he or she pays attention to the *cause* relationship between "stop" and "open". This case is simple, because the candidate of the mapping is only one, i.e. the mapping from *cause* to the "next" from Stop to Open, as shown in the figure. Another example is to establish the mapping from *instance_of* to *abstraction* between Lifts and Lift as shown in the figure, and it can be automatically decided.

As shown in these examples, the model engineers can be supported to establish the semantic mapping, in addition to detect semantic inconsistency by using inference rules on ontologies.

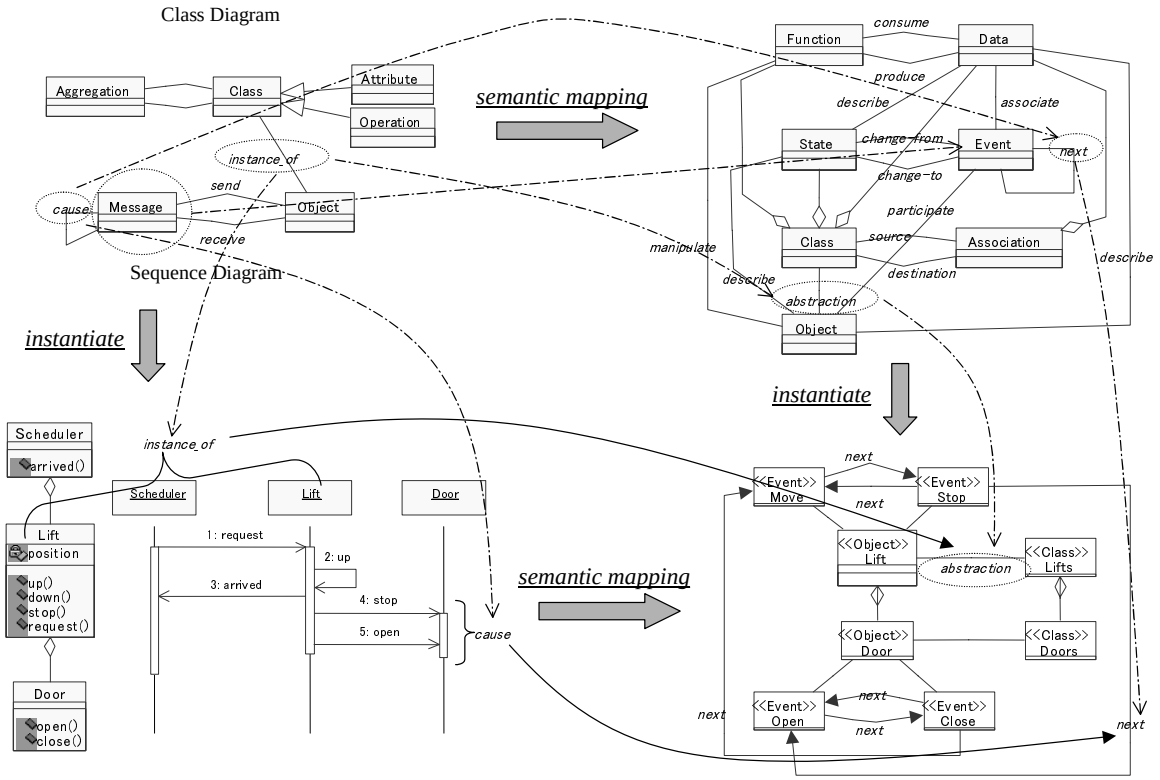
6 Research Agenda

This report discussed the relationships among domain specific models, domain ontologies, meta models and a meta model ontology, and illustrated their application. We showed a technique to give semantics to both models and meta models by using ontologies and illustrated that the logical inference rules on the ontologies could automatically detect semantic inconsistency included in the models and the meta models. We classified the ontologies into two types; domain ontology and meta model ontology, and the same technique of semantic mappings helps model engineers in providing the meaning of the models. Currently, we are elaborating the meta model ontology and developing the supporting tool for semantically inconsistency checking of models. The details of research agenda in this direction can be summarized as follows.

1. Elaborating the meta model ontology. We have diverted a simplified version of method ontology mentioned in [7] as our meta model ontology. However, we need more elaborated version rather than it. There are several excellent works to construct ontologies in the domain of information systems development and software engineering processes [10, 5, 14] and we may use their results. Another approach is the application of text-mining techniques to methodology manuals written in natural language to extract the concepts and relationships so that wide varieties of meta models can be semantically described by using them.

Meta Model

Ontology of Meta Models



Model (Class Diagram + Sequence Diagram)

Domain Ontology of Lift Control Systems

Figure 5: Meta Model Ontology and Domain Ontology

2. Supporting tool. Our inference rules are being described with Prolog because it can have application interfaces to Java like tuProlog [2]. Thus it is easier to develop GUI of the tool using Java and to extend inference rules. As a standard, description logic and its reasoning engines become more popular to specify ontologies and their inference rules. After establishing the ontologies and rules, we can shift our formalism from Prolog to description logic.

3. The supporting techniques to develop domain ontologies. Although we have a meta model of domain ontologies, developing various kind of domain ontologies of high quality by hand is a time-consuming and harder task. Adopting text mining approaches are one of the promising ones to support the development of domain ontologies [1, 9].

References

[1] KAON Tool Suite. http://kaon.semanticweb.org/.

[2] tuprolog *http://lia.deis.unibo.it/research/tuprolog/* .

[3] S. Brinkkemper, M. Saeki, and F. Harmsen. Meta-Modelling Based Assembly Techniques for Situational Method Engineering. *Information Systems*, 24(3):209 –228, 1999.

[4] S. Brinkkemper, M. Saeki, and F. Harmsen. A Method Engineering Language for the Description of Systems Development Methods. In *Lecture Notes in Computer Science (CAiSE'2001)*, volume 2068, pages 473–476, 2001.

[5] E. Falkenberg, K. Lyytinen, and A. Verrijn-Stuart, editors. *Information System Concepts: An Integrated Discipline Emerging, IFIP TC8/WG8.1 International Conference on Information System Concepts: An Integrated Discipline Emerging (ISCO-4)*.

[6] M. Gruninger and J. Lee. Ontology: Applications and Design. *Commun. ACM*, 45(2), 2002.

[7] F. Harmsen. *Situational Method Engineering*. Moret Ernst & Young Management Consultants, 1997.

[8] Haruhiko Kaiya and Motoshi Saeki. Ontology Based Requirements Analysis: Lightweight Semantic Processing Approach. In *QSIC 2005, Proceedings of The 5th International Conference on Quality Software*, pages 223–230, 2005.

[9] L. Kof. Natural Language Processing for Requirements Engineering: Applicability to Large Requirements Documents. In *Proc. of the Workshops, 19th International Conference on Automated Software Engineering*, 2004.

[10] M. Leppanen. Towards an Ontology for Information Systems Development. //http://emmsad06.idi.ntnu.no/, 2006.

[11] A. Maedche. *Ontology Learning for the Semantic Web*. Kluwer Academic Publishers, 2002.

[12] M. Saeki. Role of Model Transformation in Method Engineering. In *Lecture Notes in Computer Science (Proc. of CAiSE'2002)*, volume 2348, pages 626–642, 2002.

[13] M. Saeki, H. Horai, and H. Enomoto. Software Development Process from Natural Language Specification. In *Proc. of 11th International Conference on Software Engineering*, pages 64–73, 1989.

[14] Y. Wand. Ontology as a Foundation for Meta-Modelling and Method Engineering. *Information and Software Technology*, 38(4):281–288, 1996.