Does 'Merging DEMO Models' Satisfy the Associative Law?
Validation of Partial Models and Merge Operation

Tetsuya Suga and Junichi Iijima
Graduate School of Decision Science and Technology, Tokyo Institute of Technology, Meguro-ku, Tokyo, Japan

Keywords: Enterprise Ontology, Business Process Modeling, Formalization, Algebra, Set Theory, DEMO.

Abstract: Partial models are small models produced by splitting a large full model into meaningful conceivable-sized fragments with each separate diagram. They are commonly used when the full model is too large and shared by several people who have their own scope of interest. Those partial models are subject to being manipulated—merged for instance. This context calls for discussion in Enterprise Ontology (EO) about the capability of business process modeling languages in handling partial models and manipulations on them. There, indeed, exists a lack of researches in the methodology of EO, namely Design & Engineering Methodology for Organizations (DEMO) for formal studies on its consistency in producing partial models and merging them. It stems from a deficiency of formal semantics in the specification of the notation. By formalizing the DEMO Construction Model (CM) with a concept of well-formed models and the merge operation from a set-theoretic approach, this paper clarifies that the closedness, commutativity, and associativity are guaranteed in merging partial models of DEMO CM. An example of EU-Rent accompanies the formalizations for validation and demonstration.

1 INTRODUCTION

Since human beings began to practice the division of labor, business processes extend over several operational units within an organization or even over multiple organizations, and consist of many activities performed by different persons. In such a situation, *one overall process model can be shared by several people who have their own specialized area, hence have a different scope of interest in the model*. At the same time, as modern society has grown more sophisticated and complicated, the scope required to be covered in process models also becomes larger, sophisticated and complicated.

Enterprise Engineering (EE) is an emerging research discipline that pursues better understandings and implementations from the perspective of engineering. Among three domains[1] of EE, Enterprise Ontology studies an understanding—in other words, *modeling*—of the operation in enterprises as entirely independent of their realization and implementation. This characteristic contributes for challenging the problem in modern society.

The general trend mentioned in the first paragraph makes it necessary to take an arbitrary part of an overall model for **generating a partial model**—*splitting the large overall model into meaningful conceivable-sized fragments with each separate diagram* (also referred to as modularizing and decomposing). For example, given a business process model of a supply chain as an overall model, the sales manager hopes to have a partial model focused on processes between stores and warehouses while the procurement manager would like to focus on processes between suppliers and assembly factories (see Figure 1). Sometimes it is done for the sake of simplicity and understandability throwing away things out of interest, and sometimes for serious practical concerns that models are too large to print on a single sheet of paper. In addition to just deriving partial models, those partial models are subject to **being manipulated**—*merge, match, diff, split, slice, etc.*—for various purposes. During these manipulations, the syntactic, notational or grammatical correctness, validity, consistency, you name it on the manipulations are usually reserved by decent efforts and considerations of experienced experts.

Problem Definition and Motivation

This context reinforces discussions for each modeling

[1]Enterprise Ontology (EO), Enterprise Architecture (EA), and Enterprise Governance.

Suga, T. and Iijima, J..
Does 'Merging DEMO Models' Satisfy the Associative Law? - Validation of Partial Models and Merge Operation.
In *Proceedings of the 7th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2015) - Volume 2: KEOD*, pages 467-478
ISBN: 978-989-758-158-8

language on the handleability in handling partial models and operations on the domain of partial models, as one of the characteristics of modeling languages themselves. Desirable modeling languages should be less likely to cause inconsistency when users produce partial models and manipulate them.

The following high-school-level example may figure out the meaning of "the handleability in handling partial models and operations on the domain of partial models". Let's say *one plus two equals three* $(1+2=3)$. Suppose that the domain is the natural numbers $1,2,3,\ldots$ $(\mathbb{N})$. Then, *plus* is an operation on the domain $\mathbb{N}$ and the result *three* is a natural number. In general, if you apply the operation *plus* to two natural numbers, the result is a natural number and never become a number out of the domain (formally $\forall n,m \in \mathbb{N}, n+m \in \mathbb{N}$; closedness). You can change the order like $1+2=2+1$ (commutativeness). If you however say *one minus two equals minus one* $(1-2=-1)$, the situation has been changed. The result *minus one* is **not** a natural number. You **cannot** change the order like $1-2 \neq 2-1$ because of $1-2=-1$ and $2-1=1$. However, if you replace the domain with the integers $\ldots,-3,-2,-1,0,1,2,3,\ldots$ $(\mathbb{Z})$, the result *minus one* is an integer and thus still remains in the domain. Although you solved the trouble in closedness, you still **cannot** change the order. Therefore, whether operations have properties such as closedness and commutativeness or not depends on how you define the domain ($\mathbb{N}$, $\mathbb{Z}$, etc.) **and** the operations (addition *plus*, subtraction *minus*, multiplication *times*, division *divide*, etc.). This paper will define a domain and an operation as well.

Let us go back to the original topic. As a methodology of EE, this paper assumes and follows Design & Engineering Methodology for Organizations (DEMO). DEMO is said to be coherent, consistent, comprehensive and essential methodologies, showing an organization as a network of responsibilities and interactions. A DEMO model is a compact yet complete model representing "who is responsible for what". It focuses only on the ontological aspect of the organization, resulting in giving insight and overview of the organization. DEMO has four aspect models such as Construction Model (CM) at the top with the highest abstraction, Process Model (PM), Action Model (AM) and Fact Model (FM).

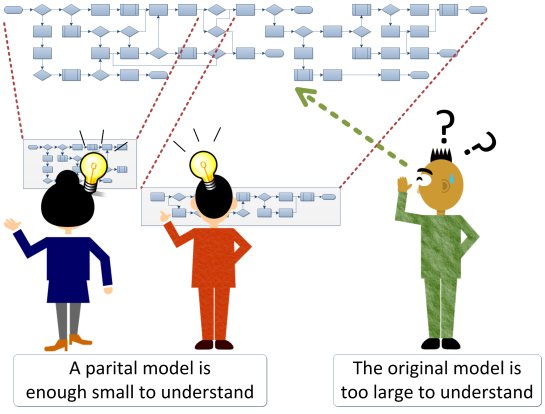

Figure 1: Image of Problem.

As DEMO is designed to be consistent, the authors feel—other DEMO users might likely agree with—solid thoughts that there is less opportunity to encounter inconsistency when generating partial models and manipulating them in DEMO models during past practical projects. *However, some questions may arise:*

> **Q1**: *What are the requirements for generating an appropriate partial model from the overall model?*
>
> **Q2**: *Is there any possibility to produce a broken model (a model which does not follow the correct syntax) when merging two partial models? i.e. closedness*
>
> **Q3**: *Does the order of merging matter?*
>
> *i.e. commutativity:* $X+Y \stackrel{?}{=} Y+X$ *and associativity:* $(X+Y)+Z \stackrel{?}{=} X+(Y+Z)$

The motivation of this study is to answer these questions with formal reasoning.

Related Works

Several papers have addressed the problem about ways to manage large models efficiently and effectively: handling partial models and manipulating them with issues of consistency. In recent works, (Enjo et al., 2010) studied this problem on UML Class Diagram, and (Liepins et al., 2012) on Web Ontology Language. (Mancioppi et al., 2012) discussed classifications for fragmentation techniques and common rationale stories on the scene.

To the best of the authors' knowledge, formalization of DEMO Construction Model (CM) has not been studied very well. Although DEMO Process Model (PM) has been formalized in Petri Net, it seems difficult to extend the formalization to CM because CM does not have the concept of states and transitions which Petri Net stands on. Although formal descriptions for the meta-model for CM have been presented on occasion, ones for CM itself are very few except the CRISP model and *crispienets* explained in (Dietz and Hoogervorst, 2014). However, there still exists a lack of research on a formalization of CM which enables to answer the questions above. Even if people share a consensus on the consistency

of DEMO model, it is not trivial and nothing guarantees the consistency.

Purpose Statement

While the lack does not matter if the scope of modeling is enough small for persons to understand, this issue will become tangibly critical especially when the scope of modeling becomes complex or large vertically (depth; granularity) and/or horizontally (width) beyond the capabilities of human beings. As studies of EE and DEMO advanced, the targets (objects) of modeling have been more practical, large, and complex–from small fictional models with a few transactions such as a pizza shop and library, to large practical models with more than ten transactions such as (Op't Land et al., 2009). Moreover, the issue will become more important when those manipulations are performed by inexperienced persons or executed automatically by computer codes. In fact, there have been some active studies to handle DEMO models and execute them such as DEMO engine (van Kervel, 2011).

The significance of this study for EE and DEMO is to provide to the notation of DEMO models a formal semantics that enables to formally define and analyze the models. Such a formal representation grounded on mathematical notations describes the properties of the models in a precise way and brings a high abstraction which help with answering the questions with context-independent explanations, unlike ambiguous natural languages and diagrams. Once the model is copied down into mathematical formulae, proves goes by themselves within mathematical laws. This is supported by soundness and completeness of logic, which insist that "anything one can prove is semantically valid" and "any argument that is semantically valid is derivable".

The purpose of this study is to examine the structure of DEMO in formalism and to reveal formal rationales behind its consistency. Developing the formal descriptions of DEMO and operations on the models will answer the questions above. The remainder of this paper is compromised as follows: Section 2 presents related studies and basics of DEMO Construction Model (CM) to compose the research methodology for this paper explained in Section 3. Section 4 describes the formal representation of DEMO CM with a concept of sub-model. Finally, Section 5 defines the merge operation on the models and reveals crucial theorems of its closedness, commutativity, and associativity as the primary contribution of this study. Section 6 discusses the result of this paper, leading to the conclusion in Section 7. From Section 4 through 6, a common example of EU-Rent accompanies the formalizations for the validation and demonstration.

2 LITERATURE REVIEW

As mentioned in section 1, mappings between the Process Model (PM) and Petri Net have been studied. However, it does not rise above the level of Process Mode because Construction Model (CM) does not have the concept of states and transitions which Petri Net stands on. Another part which advances in formalization is ones at meta-level. Many papers presented the formal description on occasion often in terms of information structure. However, there exists a lack of research on a formalization of CM which enables to answer the questions above.

Therefore, in this section, the authors present the structured literature review on formalizations of other business process models by models and approaches. This section also includes explanations about basics of DEMO CM. Combining these two reviews will work for designing research methodology for a formalization of DEMO CM.

2.1 Related Works on Formalization of Other Modeling Languages

While there exists a variety of modeling languages, they may be divided into two: flow[2] and non-flow[3] based modeling languages. In the formalism of flow based modeling languages, the most common approaches are state-transition-systems such as Petri Net including Workflow Net, graphs such as Workflow Graph, and algebra such as process algebra including CSP and general algebra of categories or sets.

These approaches are, however, of no use for formalizing DEMO Construction Model (CM) due to its non-flow based structure[4]. This point requires reviewing related works especially on non-flow based modeling languages. In this paper, the authors focus on UML Class Diagrams and UML Use Case Diagrams because these are ones of the most popular languages with advanced studies.

[2]Workflow and data flow models such as UML Sequence Diagram, UML Activity Diagram, BPMN, EPC, RAD, BPEL, YAWL, DEMO Process Model, etc.

[3]For example, UML Class Diagram, UML Use Case Diagram, DEMO Construction Model, etc.

[4]Some papers, to our surprise, still find out ways to utilize Petri Net for non-flow based diagrams such as (Baresi and Pezz, 2001) for UML Class Diagram and (Zhao and Duan, 2009) for UML Use Case Diagram.

(Shroff and France, 1997) adopted Z notation in Class Diagram for representing classes, associations, aggregations and generalization structures. (Sengupta and Bhattacharya, 2006) hired the same approach in Use Case Diagram.

(Meng and Aichernig, 2003) adopted category theory with an algebraic approach in Class Diagram and Use Case Diagram, proposing coalgebraic semantics of classes, associations, and generalization, and giving an example of consistency checking.

(Enjo et al., 2010) adopted algebraic set theory for representing classes, associations, and generalizations with discussions on techniques to keep consistency and operations (union, intersection, difference, complement) on the algebra.

(Berardi et al., 2001) adopted Description Logic (DL) in Class Diagram for representing classes, associations, aggregations, constraints and generalization.

(Klimek and Szwed, 2010) adopted temporal logic in Use Case Diagram for representing include, extends and inheritance relations with verification.

In spite of these diverse studies for UML, DEMO is short of formalizations.

2.2 DEMO Construction Model

Design & Engineering Methodology for Organizations (DEMO) is a modeling methodology for an enterprise. In an enterprise as the general term referring to a kind of collaborative activity by human beings, there exists a business referring to the function perspective and an organization referring to the construction perspective. Compared with other enterprise modeling languages, DEMO puts more emphasis on the construction perspective with the language/action perspective (LAP). It has been developed with scientific and engineering research since the 1980s led by Jan Dietz at the Delft University of Technology and other active researchers in CIAO! Network[5]. DEMO has several characteristic concepts and topics such as the standard/complete transaction pattern, levels of abstraction (business-information-data or essential-informational-documental), four aspects model (Construction Model, Process Model, Action Model, Fact Model), and the operational principle.

The operational principle in DEMO represents interactions between two parties for a transaction kind[6]

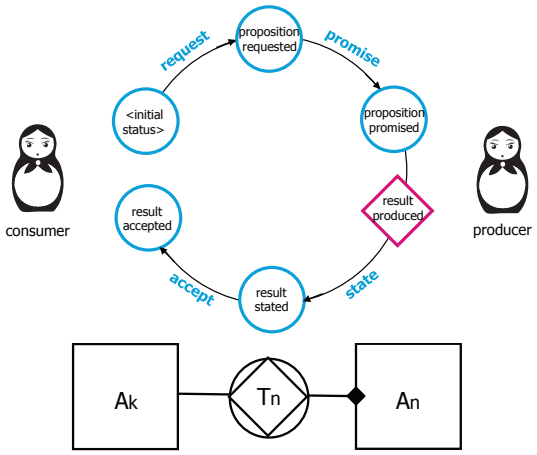

Figure 2: The Operation Principle, Excerpted from (Perinforma, 2012).

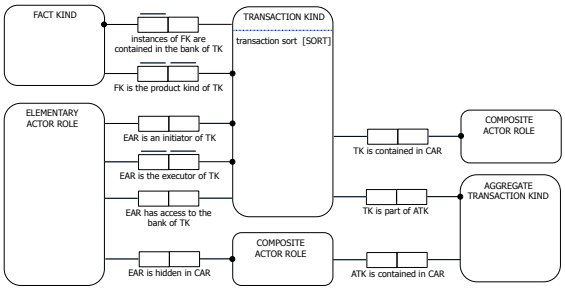

Figure 3: Meta-Model of DEMO CM, Excerpted from (Dietz, 2013).

as follows: when John wants Mary to create a desired result such as producing something or providing a service, John begins communicating to Mary with a request. The person who is responsible for the results (here Mary) provides in response to the request to answer a promise. After a certain time when Mary finishes the work, she will state that the desired result is achieved. If the person who had asked for the result (here John) accepts the result, the whole interaction will be finished. DEMO defines a transaction kind as the pattern described in the interaction between two parties, and a business as the chain of transactions.

Among the four aspect models, the Construction Model (CM) is the ontological model of the construction of organizations: the composition (inside), the environment (outside), and the structure. Specifically, they are consists of the active and passive mutual interactions among the elements in the composition and such interactions between them and the ones in the environment. The CM contains the identified actor roles

[5]http://ciaonetwork.org/

[6]In DEMO, a *transaction* and a *transaction kind* are distinguished, similar to an *object (instance)* and a *class* in object-oriented programming language such as Java. A *transaction kind* is a template, as well as a *class* is an extensible template with member variables (for representing state) and methods (or member functions; implementations of operation). For every execution, a *transaction kind* generates a new *transaction* as an instance of the *transaction kind*, as well as a *class* generates a new *object* of the *class*.

in the composition and in the environment, the identified transaction kinds between the actor roles in the scope, and between those and the actor roles in the environment. Formally, the components of the CM are shown in the form of meta model in Figure 3. The CM of an organization is represented with an Organization Construction Diagram (OCD), a Transaction Product Table (TPT), and a Bank Contents Table (BCT).

Please refer to (Dietz, 2006; Perinforma, 2012; Dietz, 2012; Dietz, 2013) for EO and DEMO.

3 RESEARCH METHODOLOGY

3.1 Choice of Approach

According to literature review on formalizations of other modeling languages, the following approaches may works: Z notation, set algebra, category algebra, Description Logic, temporal logic and Petri Net.

Z notation has been developed by the Programming Research Group (PRG) in Oxford University as a formal specification language for describing and modeling software and computer-based systems. As template-like schema notations to show the structure of specifications, It uses well-defined types (set, sequences, relations, functions) and defines operations by showing pre/post states. Z tackles the problem that large specifications reduces the readability and manageability if using mathematics alone. Besides these advantages, Z may bring the barrier of the notation. In addition, while formalizing models by Z is pretty common, there seem no tentative attempts beyond formalization to examine algebraic properties of the operations on it as also addressed in (Moreira et al., 2004). This point leads to abstaining from employing Z notation for this study.

Temporal logic is a system of rules and symbolism for coping with propositional logic with the concept of time. Since DEMO CM does not include the concept of time or sequence, temporal logic will not perform well.

Description Logics (DL) is one of the formal knowledge representation languages. It is more powerful than propositional logic and first-order predicate logic in expressiveness and decidability of reasoning problems. Despite its expressiveness, DL has very few previous studies on operations on models.

Category theory is a mathematical theory for a collection of objects and of arrows (relations) to the study of algebraic structure, favored by computer scientists. Set theory is the mathematical theory of collections of objects. Discussions on differences between categorical and set-theoretic foundations are so complicated that this paper shall reserve the explanation for other materials. Within this paper, it is enough to say that set theory focuses on objects while category theory focuses not on objects but on the relations between the objects (morphisms). Practically, category theory is highly sophisticated realm while set theory is comprehensible at the undergraduate level mathematics.

Among these formalizations, it turns out that the algebraic specification with set theory is most suited for this study due to its low barrier of the notation and concepts, and its past results for studying operations on the formalization in (Enjo et al., 2010).

3.2 Procedures

Starting the construction of formalizations, this study was conducted along with the steps following (Enjo et al., 2010)'s work on UML Class Diagrams.

1. Enumerate basic components of DEMO CM and formalize them in the mathematical language (Section 4.1).
2. Figure out relationships and requirements among the components in OCD, and then representing them using formulae (Section 4.2).
3. Define the concept of partial models in OCD with discussions of its properties (Section 4.3).
4. Define the merge operation on OCD with discussions of its remarkable theorems (Section 5).

3.3 Assumptions and Scope

For the sake of refined simplicity, this study assumes the following limitations.

Assumption 1. *Only actor roles and transaction kinds are considered, excluding aggregate transaction kinds*

Assumption 2. *Only initiator links and executor links are presented, excluding information links among three link types*

Assumption 3. *Only a single initiator link for each transaction kind*

4 FORMALIZATION OF OCD

This section prepares formal definitions of DEMO CM for this study. Along with formal definitions, this paper shows an example of EU-Rent (Figure 4) that is originally taken from (Open Management Group, 2013).

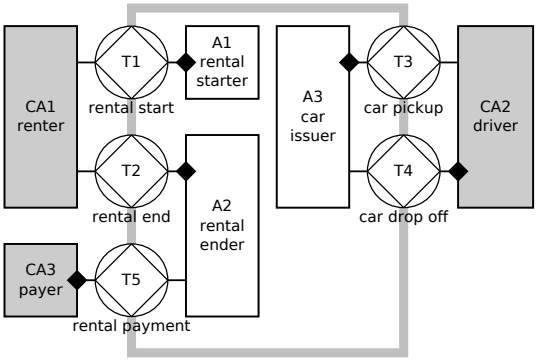

Figure 4: OCD of EU-Rent, Excerpted from (Dietz, 2012).

Please note that this paper distinguishes corresponding terms depending on whether they are used in the context of practice or that of formalization.

Practice	Formalization
partial model	sub-model
manipulation	operation
merge	union

4.1 Formal Definition of Components in DEMO CM

Definition 1 (Society)**.** A society is comprised of actor roles and transaction kinds. Let $ActorRole$ be a set of actor roles and $Transaction$ a set of transaction kinds. Then a society is described as a couple:

$$Society = \langle ActorRole, Transaction \rangle$$

Definition 2 (Actor Role Identification)**.** An actor role a has the following three mappings as its properties. In other words, actor roles are identified/characterized as a triple (3-tuple) $(f_{ARtype}(a), f_{ARno}(a), f_{ARname}(a))$:

1. ActorRole type operator

   $$f_{ARtype}\colon ActorRole \to ARtype$$

   is a mapping from an actor role to one of the types of actor role $ARtype = \{elementary, composite\}$ or in short $ARtype = \{E, C\}$.
2. ActorRole number operator

   $$f_{ARno}\colon ActorRole \to \mathbb{N}^0$$

   is a mapping from an actor role to a non-negative integer $\mathbb{N}^0$, which includes zero.
3. ActorRole name operator

   $$f_{ARname}\colon ActorRole \to \textbf{string}$$

   is a mapping from an actor role to a string[7].

[7]Let an alphabet Σ be a finite set of symbols (normally used in a natural language). A string over the alphabet Σ is a finite sequence of symbols from Σ. 'A set of strings in Σ' $\Sigma^\star$ can be recursively defined by $\Lambda \in \Sigma^\star$ and $x \in \Sigma \wedge s \in \Sigma^* \Rightarrow xs \in \Sigma^\star$(where Λ is the empty string).

Definition 3 (Transaction Kind)**.** A transaction kind t is an element of $ActorRole \times ActorRole$, where $\times$ denotes the Cartesian product (direct product). A transaction kind can be described as $t = (a, a')$. In DEMO, actor role a (and a') is called an initiator (executor), respectively.

Definition 4 (Transaction Kind Identification)**.** A transaction kind t has following four mappings as its properties. In other words, transaction kinds are identified/characterized as a quadruple (4-tuple) $(f_{Tno}(t), f_{Tname}(t), f_{Tin}(t), f_{Tex}(t))$:

1. Transaction kind number operator

   $$f_{Tno}\colon Transaction \to \mathbb{N}^+$$

   is a mapping that assigns a positive integer for each transaction kind.
2. Transaction kind name operator

   $$f_{Tname}\colon Transaction \to \textbf{string}$$

   is a mapping from a transaction kind to a string.
3. Transaction kind initiator/executor operator

   $$f_{Tin}\colon Transaction \to ActorRole$$
   $$f_{Tex}\colon Transaction \to ActorRole$$

   is a mapping from a transaction kinds to an actor role which is an initiator/executor of the transaction kind. Those operators extract the first/second operand of Cartesian product in Definition 3.

Example 1. In our society, there exists many actor roles and transaction kinds. EU-Rent is just a small part of the society. From them, modelers define the scope of interest—EU-Rent here. This situation is shown in Figure 5.

$$\begin{cases} Transaction & = \{\ldots, t_1, t_2, t_3, t_4, t_5, \ldots\} \\ ActorRole & = \{\ldots, a_0, a_1, a_2, a_3, a_4, a_5, \ldots\} \end{cases}$$

Mapping of operators are shown in Table 1.

Table 1: Mappings of the Example.

a	f_{ARtype}	f_{ARno}	f_{ARname}	t	f_{Tno}	f_{Tname}	f_{Tin}	f_{Tex}
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
a_0	C	1	renter	t_1	1	rental start	a_0	a_1
a_1	E	1	rental starter	t_2	2	rental end	a_0	a_1
a_2	E	2	rental ender	t_3	3	car pick up	a_4	a_3
a_3	E	3	car issuer	t_4	4	car drop off	a_3	a_4
a_4	C	2	driver	t_5	5	rental payment	a_2	a_5
a_5	C	3	payer					
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

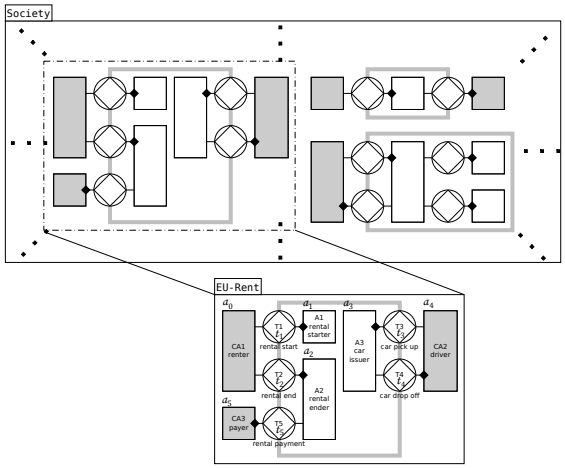

Figure 5: $Society = \langle ActorRole, Transation \rangle$.

4.2 Formal Definition of OCD

Definition 5 (Organization Construction Diagram)**.** Suppose that A is a subset of $ActorRole$ (i.e. $A \subseteq ActorRole$) and T is a subset of $Transaction$ (i.e. $T \subseteq Transaction$). Then a couple (2-tuple) $\langle A, T \rangle$ which satisfies the following conditions is called an OCD.

$$\langle A, T \rangle \in 2^{ActorRole} \times 2^{Transaction}$$

Condition 1 (Unique Actor Role Name Axiom)**.** If names of two actor roles are equal, those two actor roles are the same.

$$\forall a_i, \forall a_j \in A, (f_{ARname}(a_i) = f_{ARname}(a_j) \Rightarrow a_i = a_j)$$

Condition 2 (Unique Transaction Kind Name Axiom)**.** If names of two transaction kinds are equal, two transaction kinds are equal.

$$\forall t_i, \forall t_j \in T, (f_{Tname}(t_i) = f_{Tname}(t_j) \Rightarrow t_i = t_j)$$

Condition 3 (Numbering Axiom)**.** The executor of a transaction kind gets the same number as the transaction kind ("a very practical convention")

$$\forall t \in T, f_{ARno}(f_{Tex}(t)) = f_{Tno}(t)$$

Condition 4 (Closed OCD Axiom for Actor Role)**.** All of the actor roles responsible for a transaction kind are included in the set of actor roles of the OCD. Suppose $\langle A, T \rangle$ is an OCD, then:

$$\forall t \in T, \forall a \in ActorRole,$$
$$a \in f_{Tin}(t) \cup f_{Tex}(t) \Rightarrow a \in A$$

Condition 5 (Actor Role Participation Axiom)**.** For every actor role in the OCD, there is at least one transaction kind such that the actor role participates. Suppose $\langle A, T \rangle$ is an OCD, then:

$$\forall a \in A, \exists t \in T, a \in f_{Tin}(t) \cup f_{Tex}(t)$$

4.2.1 Related Operator

Definition 6 (Actor Role Closure for Transaction Kind)**.** Let T be a subset of transaction kinds (i.e. $T \subseteq Transaction$). Actor role closure operator

$$\gamma\colon 2^{Transaction} \rightarrow 2^{ActorRole}$$

is a mapping from a set of transaction kinds to a set of related actor roles.

$$\gamma(T) = \{a \in ActorRole \,|\, \exists t \in T, a \in f_{Tin}(t) \cup f_{Tex}(t)\}$$
$$= \bigcup_{t \in T} (f_{Tin}(t) \cup f_{Tex}(t))$$

Example 2. Let EU-Rent be $\langle A, T \rangle$, where $A = \{a_0, a_1, a_2, a_3, a_4, a_5\}$ and $T = \{t_1, t_2, t_3, t_4, t_5\}$. In this case:

$$\gamma(T) = \{a_0, a_1, a_2, a_3, a_4, a_5\}$$

4.3 Sub-society and Sub-OCD

In algebra, it is very common to single out from an arbitrary algebraic structure containing a distinguished subset, such as affine spaces and subspaces of vector spaces, sub-groups of groups, ideals and sub-rings of rings, sub-fields of fields, and so on. If intuitively generalized, these subsets are characterized to be closed under some algebraic operations defined on the structure. This section introduces the notion of sub-OCD and describes that any sub-OCDs satisfy some conditions tautologically.

4.3.1 A Family of Sub-societies

Before defining sub-OCDs, a family of sub-Societies is defined as follows.

Definition 7 (Sub-Society)**.** Given an OCD $\langle A^\circ, T^\circ \rangle$ where $A^\circ \subseteq ActorRole$ and $T^\circ \subseteq Transaction$, the OCD $\langle A^\circ, T^\circ \rangle$ can bring about 'a family of sub-Societies' D of $\langle A^\circ, T^\circ \rangle$ as below.

$$D \subseteq \{\, (A, T) | A \subseteq A^\circ, T \subseteq T^\circ \}$$

For the sake of readability, this paper hires the next expression for D which is equivalent the above.

$$D \subseteq 2^{A^\circ} \times 2^{T^\circ}$$

D is called "a family of sub-Societies" or "sub-Societies".

Please note that some members of a family of sub-Societies may *not* be an OCD, not satisfying all the conditions that OCDs must satisfy. At this point, these sub-models are "naive" in the sense that they are brought just by selecting some of the elements without considering further requirements, as claimed in the next proposition.

Proposition 1 (Partial Preservation of Sub-Societies)**.** *If a given $\langle A^\circ, T^\circ \rangle$ is an OCD, every member of the sub-Societies $D \subseteq 2^{A^\circ} \times 2^{T^\circ}$ satisfies following conditions:*

- *Condition 1: Unique Actor Role Name Axiom*
- *Condition 2: Unique Transaction Kind Name Axiom*
- *Condition 3: Numbering Axiom*

From now on, unfolded proofs for theorems, propositions, and lemmas are placed in the appendix for readability.

4.3.2 Sub-OCDs

As mentioned in the first paragraph of this chapter, one of the motivations to consider sub-OCDs is to make the sub-OCDs closed under some algebraic operations defined on the structure. The following definition imposes further conditions onto the "naive" sub-Societies.

Definition 8 (Sub-OCD)**.** Given a family of sub-Societies D of OCD $\langle A^\circ, T^\circ \rangle$, it is possible to organize 'the family of sub-OCDs' of D by selecting sub-Societies that satisfy the rest two conditions of the five conditions from D. $D^\star$ denotes the family of sub-OCDs obtained by D.

- Condition 4: Closed OCD Axiom for Actor Role
- Condition 5: Actor Role Participation Axiom

Required conditions above are the same of the original OCD. *Please note that no further considerations are needed.* From now, the rest of this section prepares equivalent expressions of the conditions to make proofs simpler.

Lemma 1 (Another Expression of Condition 4)**.** *Let $\langle A^\circ, T^\circ \rangle$ be an OCD. Then, for a family of sub-Societies $D \subseteq 2^{A^\circ} \times 2^{T^\circ}$, Condition 4 'Closed OCD Axiom for Actor Role' is the same as the following condition.*

$$\forall \langle A, T \rangle \in D, \gamma(T) \subseteq A$$

One can interpret this as meaning that for every OCD in the family of sub-OCDs, all of the actor roles that get involved with all of the transaction kinds of the OCD must be included in the actor roles of the OCD.

Lemma 2 (Another Expression of Condition 5)**.** *Let $\langle A^\circ, T^\circ \rangle$ be an OCD. Then, for a family of sub-Societies $D \subseteq 2^{A^\circ} \times 2^{T^\circ}$, Condition 5 'Actor Role Participation Axiom' is the same as the following condition.*

$$\forall \langle A, T \rangle \in D, A \subseteq \gamma(T)$$

Remark 1 (Full Preservation of Sub-OCDs). Every member of a family of sub-OCDs $D^\star \subseteq 2^{A^\circ} \times 2^{T^\circ}$ is an OCD. Among the five conditions to be an OCD, Condition 1, 2 and 3 are delivered from the characteristics of sub-Societies by Theorem 1 while the remaining condition 4 and 5 are imposed by Definition 8. Please note that "naive" sub-Societies do not always satisfy all the five conditions.

Example 3. Let us take OCD $\langle A^\circ, T^\circ \rangle$ as the whole of EU-Rent, an instance of D is:

$$D = \left\{ \begin{array}{ll} \langle \{a_0, a_1, a_2, a_4\} & , \{t_2, t_3, t_4, t_5\} \rangle, \\ \langle \{a_0, a_1, a_2\} & , \{t_1, t_2\} \rangle, \\ \langle \{a_0, a_1, a_3, a_4\} & , \{t_1, t_3\} \rangle, \\ \langle \{a_0, a_2, a_3, a_4, a_5\} & , \{t_2, t_4, t_5\} \rangle \end{array} \right\} \subseteq 2^{A^\circ} \times 2^{T^\circ}$$

Since $\langle A^\circ, T^\circ \rangle$ satisfies Condition 1, 2 and 3, every element of D satisfies these three conditions too as trivially confirmed.

Now let's check one by one whether the elements of D are sub-OCDs or not.

- $A_0 = \{a_0, a_1, a_2, a_4\}, T_0 = \{t_2, t_3, t_4, t_5\}$

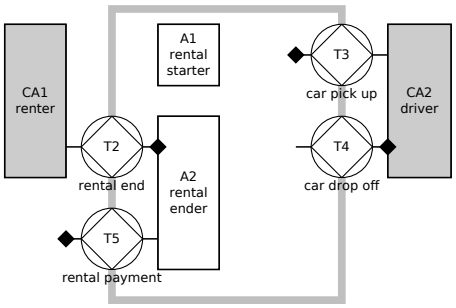

Figure 6: $\langle A_0, T_0 \rangle$.

$\langle A_0, T_0 \rangle$ is a sub-Society, but *not* a sub-OCD because $\langle A_0, T_0 \rangle$ does not satisfy Condition 4 via Lemma 1 or Condition 5 via Lemma 2 :
$\gamma(T_0) = \{a_0, a_2, a_3, a_4, a_5\} \nsubseteq A_0$
$A_0 = \{a_0, a_1, a_2, a_4\} \nsubseteq \{a_0, a_2, a_3, a_4, a_5\}$

- $A_1 = \{a_0, a_1, a_2\}, T_1 = \{t_1, t_2\}$

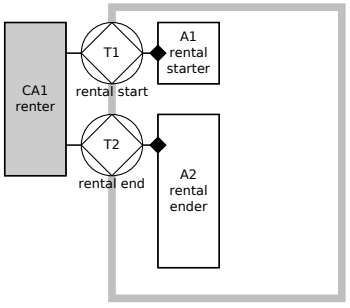

Figure 7: $\langle A_1, T_1 \rangle$.

$\langle A_1, T_1 \rangle$ is a sub-OCD.
$\gamma(T_1) = \{a_0, a_2, a_3\} \subseteq A_1$
$A_1 = \{a_0, a_1, a_2\} \subseteq \gamma(T_1)$

- $A_2 = \{a_0, a_1, a_3, a_4\}, T_2 = \{t_1, t_3\}$

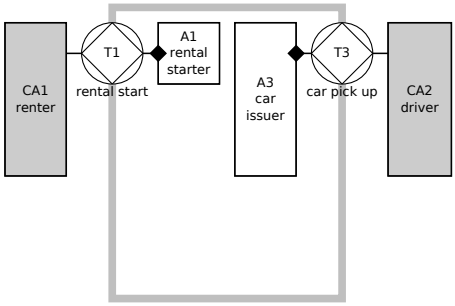

Figure 8: $\langle A_2, T_2 \rangle$.

$\langle A_2, T_2 \rangle$ is a sub-OCD.
$\gamma(T_2) = \{a_0, a_1, a_3, a_4\} \subseteq A_2$
$A_2 = \{a_0, a_1, a_3, a_4\} \subseteq \gamma(T_2)$

- $A_3 = \{a_0, a_2, a_3, a_4, a_5\}, T_3 = \{t_2, t_4, t_5\}$

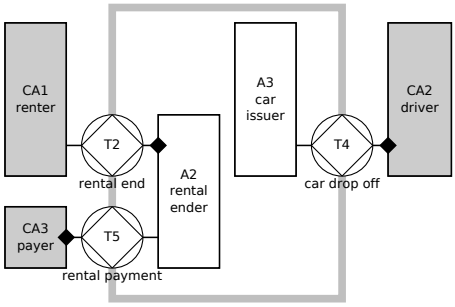

Figure 9: $\langle A_3, T_3 \rangle$.

$\langle A_3, T_3 \rangle$ is a sub-OCD.
$\gamma(T_3) = \{a_0, a_2, a_3, a_4, a_5\} \subseteq A_3$
$A_3 = \{a_0, a_2, a_3, a_4, a_5\} \subseteq \gamma(T_3)$

5 MERGE OPERATION ON SUB-OCD

In this section, we consider an operation and show that a sub-OCD is closed under the operation. An abstract notation of a merge operation is defined as follows, using the analogy of union in set theory. It is because the same actor roles are merged into one actor role, and the same transaction kinds into one transaction kind.

Definition 9 (Syntactical Merge Operation)**.** Given a OCD $\langle A^\circ, T^\circ \rangle$ and its two sub-OCDs $\langle A_X, T_X \rangle$ and $\langle A_Y, T_Y \rangle$, a syntactical merge operation

$$\nabla : (2^{ActorRole} \times 2^{Transaction}) \times (2^{ActorRole} \times 2^{Transaction}) \rightarrow (2^{ActorRole} \times 2^{Transaction})$$

is defined as follows.

$$\langle A_X, T_X \rangle \nabla \langle A_Y, T_Y \rangle = \langle A_X \cup A_Y, T_X \cup T_Y \rangle$$

Based on the definition above, there exists two important properties for the merge operation such that (1) if we apply the merge operation for two sub-OCDs from a family of sub-OCDs results in a sub-OCD from the same family, and that (2) the merge operation is commutative and associative on the family of sub-OCDs.

Theorem 1 (Closed Merge Operation)**.** *A family of sub-OCDs $D^\star \subseteq 2^{A^\circ} \times 2^{T^\circ}$ is closed under the merge operation ∇, i.e. for $\forall \langle A_X, T_X \rangle \in D^\star, \forall \langle A_Y, T_Y \rangle \in D^\star$,*

$$(\langle A_X, T_X \rangle \nabla \langle A_Y, T_Y \rangle) \in D^\star$$

Theorem 2 (Commutative and Associative Merge Operation)**.** *The merge operation ∇ is commutative and associative on a family of sub-OCDs $D^\star \subseteq 2^{A^\circ} \times 2^{T^\circ}$, i.e. for $\forall \langle A_X, T_X \rangle \in D^\star, \forall \langle A_Y, T_Y \rangle \in D^\star, \forall \langle A_Z, T_Z \rangle \in D^\star$,*

$$\langle A_X, T_X \rangle \nabla \langle A_Y, T_Y \rangle = \langle A_Y, T_Y \rangle \nabla \langle A_X, T_X \rangle$$

and

$$(\langle A_X, T_X \rangle \nabla \langle A_Y, T_Y \rangle) \nabla \langle A_Z, T_Z \rangle = \langle A_X, T_X \rangle \nabla (\langle A_Y, T_Y \rangle \nabla \langle A_Z, T_Z \rangle)$$

hold.

Proposition 2 (Units of Algebra)**.** *$\langle \emptyset, \emptyset \rangle$ and $\langle A^\circ, T^\circ \rangle$ satisfy the five conditions of OCD. $\langle \emptyset, \emptyset \rangle$ is the identity element, and $\langle A^\circ, T^\circ \rangle$ is the absorbing element (zero element). Given an OCD $\langle A^\circ, T^\circ \rangle$, for any sub-OCD $\langle A, T \rangle$ among a family of sub-OCDs $D^\star$,*

$$\langle \emptyset, \emptyset \rangle \nabla \langle A, T \rangle = \langle A, T \rangle \nabla \langle \emptyset, \emptyset \rangle = \langle \emptyset, \emptyset \rangle$$

and

$$\langle A^\circ, T^\circ \rangle \nabla \langle A, T \rangle = \langle A, T \rangle \nabla \langle A^\circ, T^\circ \rangle = \langle A^\circ, T^\circ \rangle$$

hold.

Although the proof of Proposition 2 is obtained straightforward by the definition of ∇ (see Definition 9), it is important to note that $\{\langle \emptyset, \emptyset \rangle\} \cup D^\star \cup \{\langle A^\circ, T^\circ \rangle\}$ is an algebraic structure (with the identity element and absorbing element) that is closed under ∇, based on Theorem 1, Theorem 2, and Proposition 2.

6 DISCUSSION

Finally, the authors will present the meaning of merge operation using the example of EU-Rent. According to Example 3, $\langle A_1, T_1 \rangle$, $\langle A_2, T_2 \rangle$ and $\langle A_3, T_3 \rangle$ belong to the family of sub-OCDs $D^\star$.

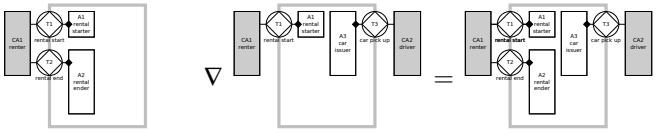

Figure 10: $\langle A_1, T_1 \rangle \nabla \langle A_2, T_2 \rangle = \langle A_4, T_4 \rangle$.

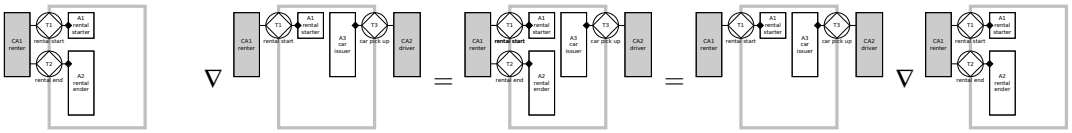

Figure 11: $\langle A_1, T_1 \rangle \nabla \langle A_2, T_2 \rangle = \langle A_4, T_4 \rangle = \langle A_2, T_2 \rangle \nabla \langle A_1, T_1 \rangle$.

Let us consider to merge $\langle A_1, T_1 \rangle$ and $\langle A_2, T_2 \rangle$ to obtain the merged model $\langle A_4, T_4 \rangle$.

$$\begin{aligned}
&\langle A_1, T_1 \rangle \nabla \langle A_2, T_2 \rangle = \langle A_1 \cup A_2, T_1 \cup T_2 \rangle \\
&= \langle \{a_0, a_1, a_2\} \cup \{a_0, a_1, a_3, a_4\}, \{t_1, t_2\} \cup \{t_1, t_3\} \rangle \\
&\quad = \langle \{a_0, a_1, a_2, a_3, a_4\}, \{t_1, t_2, t_3\} \rangle = \langle A_4, T_4 \rangle
\end{aligned}$$

Theorem 1 is illustrated as $\langle A_4, T_4 \rangle = \langle \{a_0, a_1, a_2, a_3, a_4\}, \{t_1, t_2, t_3\} \rangle \in D^\star$. The corresponding graphical representation in Figure 10 shows that the obtained $\langle A_4, T_4 \rangle$ surely looks being an OCD.

Theorem 2 is illustrated as follows. For the commutativity (shown in Figure 11),

$$\begin{aligned}
&\langle A_1, T_1 \rangle \nabla \langle A_2, T_2 \rangle \\
&= \langle \{a_0, a_1, a_2\} \cup \{a_0, a_1, a_3, a_4\}, \{t_1, t_2\} \cup \{t_1, t_3\} \rangle \\
&= \langle \{a_0, a_1, a_2, a_3, a_4\}, \{t_1, t_2, t_3\} \rangle
\end{aligned}$$

and

$$\begin{aligned}
&\langle A_2, T_2 \rangle \nabla \langle A_1, T_1 \rangle \\
&= \langle \{a_0, a_1, a_3, a_4\} \cup \{a_0, a_1, a_2\}, \{t_1, t_3\} \cup \{t_1, t_2\} \rangle \\
&= \langle \{a_0, a_1, a_2, a_3, a_4\}, \{t_1, t_2, t_3\} \rangle
\end{aligned}$$

result in the goal.

$$\langle A_1, T_1 \rangle \nabla \langle A_2, T_2 \rangle = \langle A_2, T_2 \rangle \nabla \langle A_1, T_1 \rangle$$

For the associativity,

$$\begin{aligned}
&(\langle A_1, T_1 \rangle \nabla \langle A_2, T_2 \rangle) \nabla \langle A_3, T_3 \rangle \\
&= \begin{array}{l} \langle \{a_0, a_1, a_2, a_3, a_4\}, \{t_1, t_2, t_3\} \rangle \\ \nabla \langle \{a_0, a_2, a_3, a_4, a_5\}, \{t_2, t_4, t_5\} \rangle \end{array} \\
&= \langle \{a_0, a_1, a_2, a_3, a_4, a_5\}, \{t_1, t_2, t_3, t_4, t_5\} \rangle
\end{aligned}$$

and

$$\begin{aligned}
&\langle A_1, T_1 \rangle \nabla (\langle A_2, T_2 \rangle \nabla \langle A_3, T_3 \rangle) \\
&= \begin{array}{l} \langle \{a_0, a_1, a_2\}, \{t_1, t_2\} \rangle \\ \nabla \langle \{a_0, a_1, a_2, a_3, a_4, a_5\}, \{t_1, t_2, t_3, t_4, t_5\} \rangle \end{array} \\
&= \langle \{a_0, a_1, a_2, a_3, a_4, a_5\}, \{t_1, t_2, t_3, t_4, t_5\} \rangle
\end{aligned}$$

result in the goal.

$$\begin{aligned}
&(\langle A_1, T_1 \rangle \nabla \langle A_2, T_2 \rangle) \nabla \langle A_3, T_3 \rangle \\
&= \langle A_1, T_1 \rangle \nabla (\langle A_2, T_2 \rangle \nabla \langle A_3, T_3 \rangle)
\end{aligned}$$

7 CONCLUSION

This paper shows that if an OCD $\langle A^\circ, T^\circ \rangle$ is given, one can construct D by gathering some pairs of subset of A° and T°, then construct $D^\star$ by selecting the pairs that satisfy Condition 4 and Condition 5 among D. Now $D^\star$ is a family of sub-OCDs. Indeed, a family of sub-OCDs is endowed with a beneficial characteristic which corresponds to the *well-formedness* in UML Class Diagram studied in (Enjo et al., 2010). In short, the most prominent finding of this study is that if an OCD $\langle A^\circ, T^\circ \rangle$ is given, a family of sub-OCDs of $\langle A^\circ, T^\circ \rangle$ is *well-formed* as it is and this fact is derived just from the five conditions of OCD. No further conditions and restrictions are needed in the case of OCD[8]. Here, *well-formedness* means the closedness, commutativeness, and associativeness.

Prior to explaining algebraic structure for a formalization of the DEMO CM, this paper prepared the components in DEMO with the mathematical notation and defined a family of sub-OCDs. Based on the formalization, the authors introduced the merge operation on the CM and showed that the operation conforms the closedness, commutativity, and associativity.

The questions raised in the opening section are now answered.

Q1: *What are the requirements for generating an appropriate partial model from the overall model?*

– To ensure the five conditions of OCD are enough. No further requirements are needed.

[8]This is not always the case for other modeling languages. For the case of UML Class Diagram studied in (Enjo et al., 2010), a family of *well-formed* Class Diagram is a proper (or strict) subset of Class Diagram. It means that some diagrams are *well-formed* but others are not *well-formed* even if all of the diagrams satisfy all of the specification of the notation as UML Class Diagram. Indeed, an additional condition is required to be *well-formed*.

Q2: *Is there any possibility to produce a broken model (a model which does not follow the correct syntax) when merging two partial models? i.e. closedness*

– No, merging two partial models always produces the correct model if the two satisfy the five conditions. No need to worry about producing a broken model which does not satisfy the five conditions of OCD.

Q3: *Does the order of merging matter? i.e. commutativity:* $X+Y \stackrel{?}{=} Y+X$ *and associativity:* $(X+Y)+Z \stackrel{?}{=} X+(Y+Z)$

– No, it does not matter. The equality is attained for the both formula.

Therefore, it concludes that DEMO—at least OCD—itself is consistent for the merge operation; no extra conditions are needed for DEMO OCD to be *well-formed* and to provide the closedness, commutativity, and associativity.

The future work foresees an expansion of the scope for formalization beyond the assumptions: to include aggregate transaction kinds, information links, and multiple initiator links for each transaction kind. Concurrently with the expansion, it would be worth introducing the concept of 'the scope of interest' which is regarded as the system boundary on the model.

REFERENCES

Baresi, L. and Pezz, M. (2001). On Formalizing UML with High-Level Petri Nets. In *Concurrent OOP and PN*, pages 276–304. Springer-Verlag Berlin Heidelberg.

Berardi, D., Calvanese, D., and Giacomo, G. D. (2001). Reasoning on UML Class Diagrams using Description Logic Based Systems. In *Proceedings of the KI-2001 Workshop on Applications of Description Logics (KIDLWS'01)*, pages 1–12.

Dietz, J. and Hoogervorst, J. (2014). The ψ-theory. In *TEEMs (Theories in Enterprise Engineering Memorandum)*.

Dietz, J. L. (2006). *Enterprise Ontology*. Springer Berlin Heidelberg, Berlin, Heidelberg.

Dietz, J. L. (2012). DEMO-3 Way of Modelling Way of Working (version 3.5, September 2012).

Dietz, J. L. (2013). DEMO-3 Models and Representations (version 3.6c, March 2013).

Enjo, H., Tanabu, M., and Iijima, J. (2010). A Step Toward Foundation of Class Diagram Algebra for Enterprise Service Systems. In *6th International Conference on Service Systems and Service Management, 2009. ICSSSM '09*, pages 412–417.

Klimek, R. and Szwed, P. (2010). Formal Analysis of Use Case Diagrams. *Computer Science*, 11:115–131.

Liepins, R., Cerans, K., and Sprogis, A. (2012). Visualizing and Editing Ontology Fragments with OWLGrEd. In Lohmann, S. and Pellegrini, T., editors, *the I-SEMANTICS 2012 Posters & Demonstrations Track*, pages 22–25, Graz, Austria.

Mancioppi, M., Danylevych, O., Karastoyanova, D., and Leymann, F. (2012). Towards classification criteria for process fragmentation techniques. *Lecture Notes in Business Information Processing*, 99 LNBIP(PART 1):1–12.

Meng, S. and Aichernig, B. K. (2003). Towards a Coalgebraic Semantics of UML : Class Diagrams and Use Cases. Technical report, UNU/IIST Report No. 272.

Moreira, A. M., Ringeissen, C., Déharbe, D., and Lima, G. (2004). Manipulating algebraic specifications with term-based and graph-based representations. *Journal of Logic and Algebraic Programming*, 59(1-2):63–87.

Open Management Group (2013). Semantics of Business Vocabulary and Business Rules (SBVR), V1.2 Annex G - EU-Rent Example.

Op't Land, M., Zwitzer, H., Ensink, P., and Lebel, Q. (2009). Towards a fast enterprise ontology based method for post merger integration. In *Proceedings of the 2009 ACM symposium on Applied Computing - SAC '09*, number May 2004, page 245, New York, New York, USA. ACM Press.

Perinforma, A. P. (2012). *The Essence of Organization*. Sapio Enterprise Engineering.

Sengupta, S. and Bhattacharya, S. (2006). Formalization of UML use case diagram - A Z notation based approach. In *2006 International Conference on Computing and Informatics, ICOCI '06*, pages 2–7.

Shroff, M. and France, R. B. (1997). Towards a formalization of UML class structures in Z. *Proceedings Twenty-First Annual International Computer Software and Applications Conference (COMPSAC'97)*, pages 646–651.

van Kervel, S. J. H. (2011). High Quality Technical Documentation for Large Industrial Plants Using an Enterprise Engineering and Conceptual Modeling Based Software Solution. *Advances in Conceptual Modeling. Recent Developments and New Directions*, pages 383–388.

Zhao, J. and Duan, Z. (2009). Verification of use case with Petri nets in requirement analysis. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 5593 LNCS(PART 2):29–42.

APPENDIX

Proof of Proposition 1: Partial Preservation of Sub-Societies. Let $\langle A^\circ, T^\circ \rangle$ be a given OCD. Then, the first three conditions of OCD holds for every member $\langle A, T \rangle$ in a family of sub-Societies D obtained from $\langle A^\circ, T^\circ \rangle$.

- Condition 1: Unique Actor Role Name Axiom
  Since $\langle A^\circ, T^\circ \rangle$ is an OCD and thus a Society by Definition 5, $\langle A^\circ, T^\circ \rangle$ satisfies Condition 1. Then, for any a_i and a_j in A°, if $f_{ARname}(a_i) = f_{ARname}(a_j)$ then $a_i = a_j$. Next, let $\langle A, T \rangle$ be a member of a family of sub-Societies D obtained from $\langle A^\circ, T^\circ \rangle$. Now that $A \subseteq A^\circ$ by Definition 8, for any a_i and a_j in $A \subseteq A^\circ$, if $f_{ARname}(a_i) = f_{ARname}(a_j)$ then $a_i = a_j$. It concludes that Condition 1 holds for any $\langle A, T \rangle$ in D and thus holds for a family of sub-Societies D.
- Condition 2: Unique Transaction Kind Name Axiom
  Since $\langle A^\circ, T^\circ \rangle$ is an OCD and thus a Society by Definition 5, $\langle A^\circ, T^\circ \rangle$ satisfies Condition 2. Then, for any t_i and t_j in T°, if $f_{Tname}(t_i) = f_{Tname}(t_j)$ then $t_i = t_j$. Next, let $\langle A, T \rangle$ be a member of a family of sub-Societies D obtained from $\langle A^\circ, T^\circ \rangle$. Now that $T \subseteq T^\circ$ by Definition 8, for any t_i and t_j in $T \subseteq T^\circ$, if $f_{Tname}(t_i) = f_{Tname}(t_j)$ then $t_i = t_j$. It concludes that Condition 1 holds for any $\langle A, T \rangle$ in D and thus holds for a family of sub-Societies D.
- Condition 3: Numbering Axiom
  Since $\langle A^\circ, T^\circ \rangle$ is an OCD and thus a Society by Definition 5, $\langle A^\circ, T^\circ \rangle$ satisfies Condition 3. Then, for any t in T°, $f_{ARno}(f_{Tex}(t)) = f_{Tno}(t)$ holds. Next, let $\langle A, T \rangle$ be a member of a family of sub-Societies D obtained from $\langle A^\circ, T^\circ \rangle$. Now that $T \subseteq T^\circ$ by Definition 8, for any t in $T \subseteq T^\circ$, $f_{ARno}(f_{Tex}(t)) = f_{Tno}(t)$ holds. It concludes that Condition 3 holds for any $\langle A, T \rangle$ in D and thus holds for a family of sub-Societies D.

□

Proof of Lemma 1: Another Expression of Condition 4. Let $\langle A^\circ, T^\circ \rangle$ be a given OCD, and $\langle A, T \rangle$ be an element of a family of sub-Societies D obtained from $\langle A^\circ, T^\circ \rangle$. Then, $\forall t \in T, \forall a \in ActorRole, (a \in f_{Tin}(t) \cup f_{Tex}(t) \Rightarrow a \in A)$ if and only if $\gamma(T) \subseteq A$. This is obvious by Definition 6. □

Proof of Lemma 2: Another Expression of Condition 5. Let $\langle A^\circ, T^\circ \rangle$ be a given OCD, and $\langle A, T \rangle$ be an element of a family of sub-Societies D obtained from $\langle A^\circ, T^\circ \rangle$. Then, $\forall a \in A, \exists t \in T, a \in f_{Tin}(t) \cup f_{Tex}(t)$ if and only if $A \subseteq \gamma(T)$. This is obvious by Definition 6. □

Proof of Theorem 2: Commutative and Associative Merge Operation. Let $\langle A^\circ, T^\circ \rangle$ be a given OCD. Then, suppose $\langle A_X, T_X \rangle$ and $\langle A_Y, T_Y \rangle$ are elements of a family of sub-OCDs $D^\star$ that is obtained from $\langle A^\circ, T^\circ \rangle$ via a family of sub-Societies D of $\langle A^\circ, T^\circ \rangle$. Closed in a family of sub-Societies $2^{A^\circ} \times 2^{T^\circ}$

Since $\langle A_X, T_X \rangle$ and $\langle A_Y, T_Y \rangle$ are sub-OCDs of OCD $\langle A^\circ, T^\circ \rangle$, $\langle A_X, T_X \rangle$ and $\langle A_Y, T_Y \rangle$ are off course in a family of sub-Societies $D \subseteq 2^{A^\circ} \times 2^{T^\circ}$. Thus, since $A_X \subseteq A^\circ$, $T_X \subseteq T^\circ$, $A_Y \subseteq A^\circ$, and $T_Y \subseteq T^\circ$, $A_X \cup A_Y \subseteq A^\circ$ and $T_X \cup T_Y \subseteq T^\circ$. Since $\langle A_X, T_X \rangle \nabla \langle A_Y, T_Y \rangle = \langle A_X \cup A_Y, T_X \cup T_Y \rangle$ by Definition 9, $\langle A_X \cup A_Y, T_X \cup T_Y \rangle \subseteq \{(A,T) | A \subseteq A^\circ, T \subseteq T^\circ\}$ holds. It concludes $\langle A_X, T_X \rangle \nabla \langle A_Y, T_Y \rangle \in 2^{A^\circ} \times 2^{T^\circ}$.

- Condition 4: Closed OCD Axiom for Actor Role
  Since $\langle A_X, T_X \rangle$ and $\langle A_Y, T_Y \rangle$ are sub-OCDs of OCD $\langle A^\circ, T^\circ \rangle$, $\langle A_X, T_X \rangle$ and $\langle A_Y, T_Y \rangle$ satisfy Condition 4 by Definition 8. With Lemma 1, $\gamma(T_X) \subseteq A_X$ and $\gamma(T_Y) \subseteq A_Y$ hold for $\langle A_X, T_X \rangle$ and $\langle A_Y, T_Y \rangle$ respectively. Thus, $\gamma(T_X) \cup \gamma(T_Y) \subseteq A_X \cup A_Y$. Combining this and $\gamma(T_X \cup T_Y) = \gamma(T_X) \cup \gamma(T_Y)$, $\gamma(T_X \cup T_Y) \subseteq A_X \cup A_Y$ holds. Since $\langle A_X, T_X \rangle \nabla \langle A_Y, T_Y \rangle = \langle A_X \cup A_Y, T_X \cup T_Y \rangle$ by Definition 9, Condition 4 is satisfied by Lemma 1 for $\langle A_X, T_X \rangle \nabla \langle A_Y, T_Y \rangle$.
- Condition 5: Actor Role Participation Axiom
  Since $\langle A_X, T_X \rangle$ and $\langle A_Y, T_Y \rangle$ are sub-OCDs of OCD $\langle A^\circ, T^\circ \rangle$, $\langle A_X, T_X \rangle$ and $\langle A_Y, T_Y \rangle$ satisfy Condition 5 by Definition 8. With Lemma 2, $A_X \subseteq \gamma(T_X)$ and $A_Y \subseteq \gamma(T_Y)$ hold for $\langle A_X, T_X \rangle$ and $\langle A_Y, T_Y \rangle$ respectively. Thus, $A_X \cup A_Y \subseteq \gamma(T_X) \cup \gamma(T_Y)$. Combining this and $\gamma(T_X \cup T_Y) = \gamma(T_X) \cup \gamma(T_Y)$, $A_X \cup A_Y \subseteq \gamma(T_X \cup T_Y)$ holds. Since $\langle A_X, T_X \rangle \nabla \langle A_Y, T_Y \rangle = \langle A_X \cup A_Y, T_X \cup T_Y \rangle$ by Definition 9, Condition 5 is satisfied by Lemma 2 for $\langle A_X, T_X \rangle \nabla \langle A_Y, T_Y \rangle$.

□

Proof of Theorem 2: Commutative and Associative Merge Operation. Let $\langle A^\circ, T^\circ \rangle$ be a given OCD. Then, suppose $\langle A_X, T_X \rangle$, $\langle A_Y, T_Y \rangle$, and $\langle A_Z, T_Z \rangle$ are elements of a family of sub-OCDs $D^\star$ that is obtained from $\langle A^\circ, T^\circ \rangle$ via a family of sub-Societies D of $\langle A^\circ, T^\circ \rangle$. Now the commutativity and associativity are confirmed as below.

Commutativity:

$$
\begin{aligned}
& \langle A_X, T_X \rangle \nabla \langle A_Y, T_Y \rangle \\
= \; & \langle A_X \cup A_Y, T_X \cup T_Y \rangle \\
= \; & \langle A_Y \cup A_X, T_Y \cup T_X \rangle \quad \because \text{commutative law of } \cup \\
= \; & \langle A_Y, T_Y \rangle \nabla \langle A_X, T_X \rangle
\end{aligned}
$$

Associativity:

$$
\begin{aligned}
& (\langle A_X, T_X \rangle \nabla \langle A_Y, T_Y \rangle) \nabla \langle A_Z, T_Z \rangle \\
= \; & \langle A_X \cup A_Y, T_X \cup T_Y, \rangle \nabla \langle A_Z, T_Z \rangle \\
= \; & \langle (A_X \cup A_Y) \cup A_Z, (T_X \cup T_Y) \cup T_Z \rangle \\
= \; & \langle A_X \cup (A_Y \cup A_Z), T_X \cup (T_Y \cup T_Z) \rangle \quad \because \text{associative law of } \cup \\
= \; & \langle A_X, T_X \rangle \nabla \langle A_Y \cup A_Z, T_Y \cup T_Z, \rangle \\
= \; & \langle A_X, T_X \rangle \nabla (\langle A_Y, T_Y \rangle \nabla \langle A_Z, T_Z \rangle)
\end{aligned}
$$

□