USING FORMAL CONCEPT ANALYSIS FOR VERIFICATION OF PROCESS – DATA MATRICES IN CONCEPTUAL DOMAIN MODELS

Jonas Poelmans[1], Guido Dedene[1,2], Monique Snoeck[1], Stijn Viaene[1,3]
[1] KULeuven, Faculty of Business and Economics, Naamsestraat 69, 3000 Leuven, Belgium
[2] Universiteit van Amsterdam Business School, Roeterstraat 11, 1018 WB Amsterdam, The Netherlands
[3] Vlerick Leuven Gent Management School, Vlamingenstraat 83, 3000 Leuven, Belgium
{jonas.poelmans, monique.snoeck, guido.dedene}@econ.kuleuven.be
stijn.viaene@vlerick.be

ABSTRACT
One of the first steps in a software engineering process is the elaboration of the conceptual domain model. In this paper, we investigate how Formal Concept Analysis can be used to formally underpin the construction of a conceptual domain model. In particular, we demonstrate that intuitive verification rules for process-data matrices can be formally grounded in FCA theory. As a case study, we show that the well-formedness rules from MERODE are isomorphic to the clustering rules in Formal Concept Analysis, and that the relationships in the class diagram are isomorphic to the subconcept-superconcept relationship in FCA.

KEY WORDS
Formal Concept Analysis, MERODE, conceptual domain modeling, OOSSADM, CRUD

1. Introduction

The complexity of most information systems is caused by the complexity of the reality they have to deal with and statements about the required functionality always have some underlying assumption about the real world. Therefore, it is useful to build a real world model prior to the development of an information system [1]. High quality conceptual models are critical to the success of system development efforts [2].

Unfortunately, developers often encounter problems while elaborating the business domain model [3]: inconsistencies arise between static and dynamic schemas, object types are missing in the business domain model, the business domain model contains errors, etc. Quality has been identified as one of the main topics in current conceptual modeling research [4]. Despite this importance, algorithmic approaches to assure conceptual model quality are virtually nonexistent [5]. In [6], the authors suggested to use CRUD-matrices to analyze consistency in conceptual models. However, this topic was only briefly discussed. Can we mathematically ground this type of analysis? Can we find an algorithmic approach to detect missing object types? Can we benefit from an algorithmic method for enforcing consistency in business models? Can we mathematically analyze completeness of models?

In this paper, we explore the possibilities of using a technique known as Formal Concept Analysis (FCA) [7, 8] for mathematically underpinning the construction and analysis of conceptual models. FCA arose twenty-five years ago as a mathematical theory [9]. In the domain of software engineering, FCA has typically been applied to support software maintenance. It has been used for reorganizing existing class hierarchies and for refactoring and modifying existing code [10, 11]. FCA has also been used for identifying class candidates in legacy code [12]. In requirements analysis, FCA has been used to identify class candidates in use case descriptions [13, 14] and to reconcile descriptions written by different stakeholders using controlled vocabulary and grammar [15, 16]. More recently FCA has also been used in combination with ontology. Cimiano investigates how FCA and ontologies may complement each other from an application point of view [17]. Bain applies FCA to identify structure in theories [18]. Within the area of design, FCA has been applied to classes and methods [19]. However it has never been applied to the earlier stage of conceptual modeling. In this paper we apply FCA to the combination of object types and processes to validate the relationships captured by the class diagram and to identify missing object types.

The remainder of this paper is composed as follows. In section 2, we introduce the research question. As FCA is in essence a matrix-technique, we discuss the three most frequently used matrix techniques in conceptual domain modeling: the CRUD-matrix from Information Engineering, the entity-event table from OOSSADM and the object-event table from MERODE. Subsequently we discuss how FCA could be used as formal foundation for well-formedness rules for process-data matrices in conceptual modelling. In section 3, we introduce the pivotal notions of FCA theory. As MERODE is the only method that defines well-formedness rules for a CRUD-like matrix, in section 4, we discuss the essentials of MERODE. In section 5, the close relationship between FCA and these well-formedness rules is investigated. Section 6 concludes the paper.

2. Research Question

In this section, we elaborate on the three most frequently used matrix techniques in object-oriented conceptual domain modeling.

The Create, Read, Update and Delete (CRUD)-matrix was initially introduced in information engineering by Martin [20]. The purpose of this matrix is to illustrate the relationships between objects and the processes in which they participate. A process may either create, read, update or delete an object.

The Object-Oriented Structured Systems Analysis and Design Methodology (OOSSADM) builds on the Jackson Systems Development approach [2]. In this approach, entities impose sequence constraints on business events by means of a sequence diagram. As events can appear in the sequence diagrams of multiple entities, Robinson [21] introduced the notion of Entity-Event matrix to capture the entities that are affected by each business event.

Model driven, Existence dependency Relation, Object oriented DEvelopment (MERODE) is an object-oriented analysis and design methodology [22, 23] and is complementary to UML [24], in that it offers a precise and computationally complete methodology. MERODE represents an information system through the definition of business events, their effect on enterprise objects and the related business rules.

Similarly as in OOSSADM, business events are identified as independent concepts, with an object-event table defining which types of objects are affected by which types of events. Each object type has a method for each event type in which it may participate. Such method implements the object's creation, its state changes (i.e. changes to attribute values) or its deletion as the consequence of an event of the corresponding type.

The events in OOSSADM and in MERODE can be considered as elementary processes that have an effect (create, modify or update) on at least one, but possibly more object types. Hence, in its most simple form, these three tables indicate which objects participate in which elementary processes[1]. A convenient way for representing such table is by a cross table, which is a rectangular table of which the rows are labeled by the objects, the columns labeled by events and a cross in an entry indicates that the corresponding object participates in the corresponding event.

One major difference between the three approaches is the way a table is filled. In all three approaches, the table is initially filled on the basis of classical analysis: interviews with key users and logical reasoning. Subsequently, the table should be verified against some quality criteria. Typically existing criteria will attempt to identify missing rows, columns or crosses by means of general and intuitive "rules of thumb" such as: for each object type, there should be at least one process that creates objects of that type; each process should at least read some data, etc. Of the three modeling approaches, MERODE is the only approach that defines formal criteria for the completeness and well-formedness of the table.

The CRUD-matrix, Entity-Event and Object-Event table have a clear relationship with a single-valued formal context from FCA (see section 3): object or entity types in a conceptual model can be mapped to "objects" in FCA and events can be mapped to "attributes" in FCA. Formal Concept Analysis is a recent mathematical technique that can be used as an unsupervised clustering technique. The starting point of the analysis is a table consisting of rows (i.e. objects), columns F (i.e. attributes) and crosses (i.e. relationships between objects and attributes).

The goal of the research is to investigate the possibilities of using FCA as grounding theory assisting in the development of conceptual domain models. The core contributions of this paper are as follows. First, we show that conceptual modeling can be considered as an application of FCA. FCA provides a sound mathematical foundation for assisting modelers in the elaboration of conceptual domain models. Moreover, it provides a formal underpinning for the central notion of concept. Second, we show that the well-formedness rules from MERODE - obtained by reasoning on the semantics of existence dependency, on common sense reasoning and on process algebra considerations – are an inherent part of the FCA lattice construction algorithm. Whereas in MERODE, the consistency requirements were statically modeled as meta frame for conceptual models, FCA provides an algorithm that automatically verifies whether the association matrix and existence dependency graph are correct and consistent. Starting from an Object-Event table that is well-formed according to the MERODE-rules, we apply the clustering principles of FCA and demonstrate that FCA comes up with a ordering of concepts that is isomorphic to the ordering imposed by the existence dependency relationship used in MERODE. Hence, we can postulate that FCA offers a theoretical foundation for the consistency rules of MERODE. Reversely, a principal result is that it is possible in conceptual object-oriented analysis to obtain a concept lattice by using the MERODE rules. In this way we demonstrate that FCA is a valid instrument for the formal underpinning of matrix verification techniques [23].

3. FCA essentials

Formal Concept Analysis is a recent mathematical technique that can be used as an unsupervised clustering technique. Objects participating in the same set of events are grouped in concepts. The starting point of the analysis is a table consisting of rows M (i.e. objects), columns F (i.e. attributes) and crosses $T \subseteq M \times F$ (i.e. relationships between objects and attributes).

The mathematical structure used to reference such a cross table is called a formal context (M, F, T).

[1] In the remainder of this paper, we will call these elementary processes "events". It should however be noted that these "events" do not only symbolise the initiating trigger, but also the processing that is activated as response to the event.

Table 1. Example of a formal context

	enter	leave	acquire	classify	borrow	renew	return	sell	lose
Member	X	X			X	X	X		X
Book			X	X	X	X	X	X	X
Loan					X	X	X		X

An example of a cross table is displayed in Table 1. In the latter, objects are related (i.e. the crosses) to a number of events (i.e. the attributes); here an object is related to an event if the object participates in the event. Given a formal context, FCA then derives all concepts from this context and orders them according to a subconcept-superconcept relation. This results in a line diagram (a.k.a. lattice).

The notion of concept is central to FCA. The way FCA looks at concepts is in line with the international standard ISO 704, that formulates the following definition: A concept is considered to be a unit of thought constituted of two parts: its extension and its intension [7. 8]. The extension consists of all objects belonging to the concept, while the intension comprises all attributes shared by those objects. Typically, one would think here about informational attributes but one can just as well consider behavioral attributes such as reaction to events or participation in processes. So let us illustrate the notion of concept of a formal context using the data in Table 1. For a set of objects $O \subseteq M$, the events that are common to all objects o in the set O can be identified, written $\sigma(O)$, via:

$$A = \sigma(O) = \{ f \in F \mid \forall o \in O : (o, f) \in T \}$$

Take for example the set $O \subseteq M$ consisting of objects Member, Book and Loan. This set O of objects is closely connected to a set A consisting of the attributes "borrow", "renew", "return" and "lose", being the events shared by the objects in O. That is:

σ({Member, Book, Loan}) = {borrow, renew, return, lose}

Reversely, for a set of attributes A, we can define the set of all objects that share all attributes in A:

$$O = \tau(A) = \{ i \in M \mid \forall f \in A : (i, f) \in T \}$$

If we take as example the set of events of Loan, namely {borrow, renew, return, lose}, we get to the set $O \subseteq M$ consisting of the objects Member, Book and Loan. That is to say:

τ({borrow, renew, return, lose}) = {Member, Book, Loan}

As one can see, there is a natural relationship between O as the set of all objects sharing all attributes of A, and A as the set of all attributes that are valid descriptions for all the objects contained in O. Each such pair (O, A) is called a formal concept (or concept) of the given context. The set $A = \sigma(O)$ is called the intent, while $O = \tau(A)$ is called the extent of the concept (O, A).

Notice that concepts are always maximal in the sense that the set O contains *all* objects that share the attributes of A and that A contains *all* shared attributes of the objects in O.

Moreover, there is a natural hierarchical ordering relation between the concepts of a given context that is called the subconcept-superconcept relation.

$$(O_1, A_1) \subseteq (O_2, A_2) \Leftrightarrow (O_1 \subseteq O_2 \wedge A_2 \subseteq A_1)$$

A concept $d = (O_1, A_1)$ is called a subconcept of a concept $e = (O_2, A_2)$ (or equivalently, e is called a superconcept of a concept d) if the extent of d is a subset of the extent of e (or equivalently, if the intent of d is a superset of the intent of e). For example, the concept with intent "enter," "leave," "lose," "return," "renew," and "borrow" is a subconcept of the concept with intent "lose," "return," "renew," and "borrow." With reference to Table 1, the extent of the latter is composed of object types Loan, Member and Book, while the extent of the former is composed of object type Member.

The set of all concepts of a formal context combined with the subconcept-superconcept relation defined for these concepts gives rise to the mathematical structure of a complete lattice, called the concept lattice $\beta(M, F, T)$ of the context. The latter is made accessible to human reasoning by using the representation of a (labeled) line diagram. The line diagram in Figure 1, for example, is a compact representation of the concept lattice of the formal context abstracted from Table 1. The circles or nodes in this line diagram represent the formal concepts. It displays only concepts that describe objects and is therefore a subpart of the concept lattice. The shaded boxes (upward) linked to a node represent the attributes used to name the concept. The non-shaded boxes (downward) linked to the node represent the objects used to name the concept. The information contained in the formal context of Table 1 can be distilled from the line diagram in Figure 1 by applying the following reading rule: An object g is described by an attribute m if and only if there is an ascending path from the node named by g to the node named by m. For example, Member is described by the attributes "enter", "leave", "lose", "return", "renew" and "borrow".

Retrieving the extension of a formal concept from a line diagram such as the one in Figure 1 implies collecting all objects on all paths leading down from the corresponding node. In this example, the extension

associated with the upper node is {Loan, Book, Member}. To retrieve the intension of a formal concept one traces all paths leading up from the corresponding node in order to collect all attributes. In this example, the second concept in row two is defined by the attributes "sell," "classify," "acquire," "lose," "renew," "return," and "borrow." The top and bottom concepts in the lattice are special. The top concept contains all objects in its extension. The bottom concept contains all attributes in its intension. A concept is a subconcept of all concepts that can be reached by travelling upward. This concept will inherit all attributes associated with these superconcepts. In our example, the first node on the second row with extension {Member} is a subconcept of the top node with extension {Loan, Member, Book}.

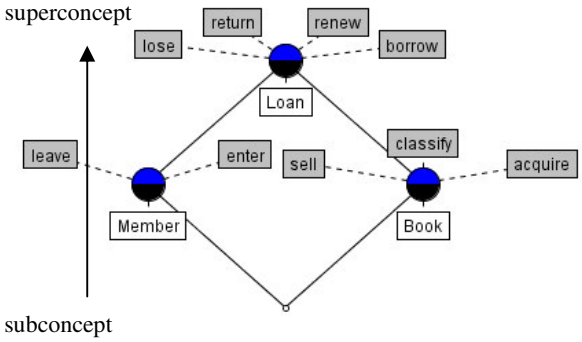

Fig. 1. Line diagram corresponding to the context from Table 1

In FCA, the concept generated by an object type P is defined as $\gamma(P) = (\tau(\sigma(P)), \sigma(P))$ and the concept generated by an event type b as $\lambda(b) = (\tau(b), \sigma(\tau(b)))$. In the line diagram, the nodes are labelled by the object types which generate the corresponding concept C. These are called the own object types of the concept C.

4. MERODE essentials

The MERODE methodology entails the notion of existence dependency, which superimposes a lattice structure (not to be confused with inheritance hierarchies) on objects. The concept of existence dependency (ED) is based on the notion of the "life" of an object. The life of an object is the span between the point in time of its creation and the point in time of its end. Existence dependency is defined at two levels: at the level of object types or classes and at the level of object occurrences. The existence dependency relation is a partial ordering on objects and object types which is defined as follows.

Definition 1 (Existence Dependency): Let P and Q be object types. P is existence dependent on Q (notation: $P \leftarrow Q$) if and only if the life of each occurrence p of type P is embedded in the life of one single and always the same occurrence q of type Q. p is called the dependent object, (P is the dependent object type) and is existence dependent on q, called the master object (Q is the master object type).

The result is that the life of the existence dependent object cannot start before the life of its master. Similarly, the life of an existence dependent object ends at the latest at the same time that the life of its master ends.

The notion of existence dependency is similar to the notion of weak entity as introduced by Chen and the notion of master entity from OOSSADM. In the ER-notation [25] we can use the notion of a weak entity to denote an existence dependent object type since the existence of a weak entity depends on the existence of the other entities it is related to by means of a weak relationship [25]. Existence dependency is equivalent to the notion of a weak relationship that is in addition mandatory for the weak entity type.

MERODE requires all objects in the conceptual model to be related through existence dependency relationships only. The class diagram can therefore be represented as an existence dependency graph.

Definition 2 (Existence Dependency Graph): Let $\mathcal{M}$ be the set of object types in the conceptual schema. The existence dependency graph (EDG) is a relation $\leftarrow$ which is a bag[2] over $\mathcal{M} \times \mathcal{M}$ such that $\leftarrow$ satisfies the following restrictions:

1) An object type is never existence dependent on itself:

$$\forall P \in \mathcal{M}: (P,P) \notin \leftarrow$$

2) Existence dependency is acyclic. This means that:

$$\forall n \in \mathbb{N}, n \geq 2, \forall P_1, P_2, ..., P_n \in \mathcal{M}:$$
$$(P_1,P_2), (P_2,P_3),..., (P_{n-1},P_n) \in \leftarrow \Rightarrow (P_n,P_1) \notin \leftarrow$$

$\twoheadleftarrow$ is the non-reflexive transitive closure of $\leftarrow$:

$\twoheadleftarrow \subseteq \mathcal{M} \times \mathcal{M}$ such that

1) $\forall P, Q \in \mathcal{M}: (P,Q) \in \leftarrow \Rightarrow (P,Q) \in \twoheadleftarrow$
2) $\forall P, Q, R \in \mathcal{M}: (P,Q) \in \leftarrow$ and $(Q,R) \in \twoheadleftarrow$ $\Rightarrow (P,R) \in \twoheadleftarrow$

In practice, MERODE also demands that the EDG is fully connected.

Definition 3 (Object Event Table): The object-event table is a table with one row for each object type and one column for each event type. Each cell contains either a blank or a 'X', which stands for "participates in event". Let A be the universe of relevant event types. Then $\mathcal{T} \subseteq \mathcal{M} \times A \times \{' ', 'X'\}$ such that $\forall P \in \mathcal{M}, \forall a \in A$:

$(P,a,' ') \in \mathcal{T}$ or $(P,a,'X') \in \mathcal{T}$

$\forall P \in \mathcal{M}: x(P) = \{a \in A \mid (P,a,'X') \in \mathcal{T}\}$

$A = \cup \{x(P) \mid P \in \mathcal{M}\}$

The OET is drawn as a matrix containing one row for each object type and one column for each event type. An 'X' on a row-column point of intersection indicates that

2. Bags can contain the same element more than once (as opposed to sets).

this particular event type is an element of x(P) (the alphabet of P), where P is the object type corresponding to the row.

In MERODE, a multiple of well-formedness rules for object-oriented conceptual models are defined. These rules were elaborated based on reasoning on model quality (completeness and consistency), on object life cycles and the formalization by means of process algebra [23, 26]. We now discuss four of these rules that are relevant for this paper. It should be noted that we take the MERODE-rules as such and do not aim at motivating these rules in this paper. For a motivation, the interested reader is referred to [22, 23].

Rule 1: the relevant life of a domain object type has a certain duration that can be delimited by two events: one event when the object enters the domain of interest and one event when the object leaves the domain of interest. In other words, each object type should participate in at least two event types: one for its creation and one for its ending. For the object-event table, this means that each row should contain at least two crosses.

Rule 2: each identified event type must be relevant for at least one object type. For the object-event table, this means that on each column there is at least one row with a cross.

Rule 3 (propagation rule): a master object type is always involved in all event types in which one of its dependent object types participates. For example, a state change of a loan, e.g. because of the return of the book, automatically implies a state change of the related book and member: the book is back on shelf and the member has one copy less in loan. Therefore, if P is existence dependent of Q, the alphabet of P must be a subset of the alphabet of Q. This is called the propagation rule: $P \leftarrow Q \Rightarrow x(P) \subseteq x(Q)$.

Rule 4 (contract rule): the contract rule says that when two object types share two or more event types, the common event types must be in the alphabet of one or more common existence dependent object types:

$\forall P, Q \in \mathcal{M} : \#(x(P) \cap x(Q)) \geq 2$ and $\neg(x(P) \subseteq x(Q)$ or $x(Q) \subseteq x(P)) \Rightarrow \exists R_1, \ldots R_n \in \mathcal{M}: \forall i \in \{1,\ldots,n\}: R_i \leftarrow P,Q$ and $x(R_1) \cup \ldots \cup x(R_n) = x(P) \cap x(Q)$

Consequence: $x(P) \subseteq x(Q) \Rightarrow P \leftarrow Q$

Notice that in MERODE the contract rule is only applicable in case of two or more common event types and that at least one of these must create and another one must end the existence dependent object types. If there is only one common event type MERODE does not require the definition of an extra object type, because an object type requires at least two event types. The argument for two events is only based on the fact that a life cycle requires a start and an end.

5. FCA and the object-event matrices

As explained before, the well-formedness rules from MERODE were obtained by reasoning on the semantics of existence dependency, on common sense reasoning and on process algebra considerations. We reformulate each of the MERODE-rules in FCA terms. The existence of rules 1 and 2 can easily be motivated in FCA as well: objects without attributes and attributes (events) that belong to no objects can be considered as incompletenesses in a conceptual model. For rule 3 it appears that the principle of propagation of events along existence dependency paths yields the result that the FCA and EDG lattice are to a large extent isomorphic. In other words, by imposing the well-formedness rules from MERODE, we obtain a concept lattice in conceptual object-oriented analysis. Reversely, the subconcept-superconcept relationship of FCA turns out to be isomorphic to the existence dependency relationship. Finally, although rule 4 (contract rule) can be formulated in FCA terms, FCA offers no substantiation for the existence of this rule. This could be a reason to revise that rule in MERODE.

5.1 Object-event table with empty row column

Consider the object-event table displayed in Table 2. The lattice corresponding to Table 2 is displayed in Figure 2. This table has an empty row, which means there is an object type R that is involved in no event type at all. In FCA, this implies that the object type R is part of the extent of the top concept of the corresponding lattice. This top concept has an empty intent. In other words, there is an object type that has no attributes: it participates in no events.

This table also has an empty column, which means there is an event type e that involves no object type at all. In FCA, this implies that the event type e is part of the intent of the bottom concept of the lattice. This bottom concept has an empty extent. In other words, there is an attribute that belongs to no object.

Table 2. Object-event table with empty row and column

	a	*b*	*c*	*d*	*e*
P		X	X	X	
Q	X	X	X		
R					

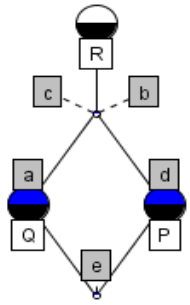

Fig. 2. Lattice corresponding to Table 2

MERODE considers these two cases as modeling anomalies. Rule 1 demands that each object type participates in at least two event types, one for the creation of objects and one for the deletion of objects. So,

empty rows are not allowed. This MERODE rule can be reformulated in FCA terms as follows.

Rule 1 in FCA terms:

Let P be an object type, L a concept and m, n events:

$Given : (M, F, T)$

$\forall P \in M :$

$\exists L \in \beta(M, F, T) : P \in ext(L) \wedge \exists m, n : m, n \in \text{int}(L) \wedge m \neq n$

MERODE also demands that each event type is relevant for at least one object type, so empty columns are not allowed. This MERODE rule can be reformulated in FCA terms as follows.

Rule 2 in FCA terms:

$Given : (M, F, T)$

$\forall a \in F : \exists L \in \beta(M, F, T) : ext(L) \neq \varnothing \wedge a \in \text{int}(L)$

5.2 Propagation rule and subconcept-superconcept relation

In this section we demonstrate that because of the propagation rule, every object in the EDG generates a tuple $\varepsilon(P) = (P^*, \text{x}(P))$, where P^* is the set of all masters of P and $\text{x}(P)$ is the alphabet of P. Then $\varepsilon(P)$ matches with the concept in the FCA line diagram with label P and moreover, if P is existence dependent on Q, then there is an upward path in the FCA lattice from the node with label Q to the node with label P.

Definition 3: Let $\mathcal{E}(P)$ be defined as follows: $\mathcal{E}(P) = (P^*, x(P))$, with $P^* = \{X \mid (P,X) \in \leftarrow\} \cup \{P\}$

Theorem 3: $\mathcal{E}(P)$ is a concept in FCA

Proof:

$\mathcal{E}(P) = (P^*, \text{x}(P))$

To be proven: $P^* = \tau(\text{x}(P)) = \{Q \in M \mid \forall\, e \in \text{x}(P) \mid e \in \text{x}(Q)\}$

$Q \in P^*$

$\Leftrightarrow (P,Q) \in \leftarrow$

$\Leftrightarrow \text{x}(P) \subseteq \text{x}(Q)$ (because of propagation rule)

$\Leftrightarrow \forall\, e \in \text{x}(P)$: $e \in \text{x}(Q)$

$\Leftrightarrow Q \in \tau(\text{x}(P))$

QED

Theorem 4: if $P \leftarrow Q$ then $\mathcal{E}(P)$ is a superconcept of $\mathcal{E}(Q)$

Proof

$\mathcal{E}(P) = (\{P\} \cup \{X \mid (P,X) \in \leftarrow\},\ \text{x}(P))$

$\mathcal{E}(Q) = (\{Q\} \cup \{Y \mid (Q,Y) \in \leftarrow\},\ \text{x}(Q))$

(1):

$P \leftarrow Q$

$\Rightarrow (Q \leftarrow Y \Rightarrow P \leftarrow Y)$

$\Rightarrow \{X \mid (P,X) \in \leftarrow\} \supseteq \{Y \mid (Q,Y) \in \leftarrow\}$

$\Rightarrow (\{P\} \cup \{X \mid (P,X) \in \leftarrow\}) \supseteq (\{Q\} \cup \{Y \mid (Q,Y) \in \leftarrow\}$

(2):

$\text{x}(P) \subseteq \text{x}(Q)$ (because of propagation rule)

(1) + (2) $\Rightarrow \mathcal{E}(P)$ is a superconcept of $\mathcal{E}(Q)$

QED.

Theorem 5: if $\gamma(P)$ is a superconcept of $\gamma(Q)$ in FCA then $P \leftarrow Q$ in MERODE

Proof

$\gamma(P) = (\tau(\sigma(P)), \sigma(P))$

$\gamma(Q) = (\tau(\sigma(Q)), \sigma(Q))$

$\Rightarrow \text{int}(\gamma(P)) \subseteq \text{int}(\gamma(Q))$

$\Rightarrow \text{x}(P) \subseteq \text{x}(Q)$

$\Rightarrow P \leftarrow Q$

QED

Consider for example the object-event table in Table 1. The set of events in which Loan participates is a subset of the events in which Member participates. In MERODE, Loan is said to be existence dependent of Member. In FCA, the concept generated by Loan is a superconcept of the concept generated by Member. As a consequence, there is an upward leading path from the node with label Member to the node with label Loan in the FCA lattice (see Figure 1).

5.3 FCA and the contract rule

In this section we show how the object-event lattice may help in identifying potentially missing object types. If the object-event lattice contains a concept with two or more own event types in its intent and zero own object types in its extent, then we have a node in the line diagram with attribute labels, but without object type labels attached to the node. In MERODE, this indicates a situation where the object types one level lower in the lattice share at least two events that do not appear in the alphabet of one or more common existence dependent object types. This situation is in contradiction with the contract rule.

Theorem 6: If a lattice $\beta(O, A, I)$ contains a concept C such that $own(\text{ext}(C)) = \varnothing$ and $own(\text{int}(C)) \geq 2$, then an object type is missing in the object-event table according to the MERODE contract rule.

Proof:

$(C \in \beta(O, A, I)) \wedge (ext(C) = \varnothing) \wedge |\text{int}(C)| \geq 2$

(1)$\{L \mid L \in \beta(O, A, I) \wedge L \subset C\} = \varnothing$

$\Rightarrow \exists a \in A : \{P \mid Y \in \beta(O, A, I) \wedge P \in ext(Y) \wedge a \in \text{int}(Y)\} = \varnothing$

$\Rightarrow$ violation of Rule 2

(2)$\exists P, Q \in O : \exists L, R \in \beta(O, A, I) \wedge L \neq R \wedge L \subset C \wedge R \subset C \wedge P \in ex$

$\Rightarrow \text{int}(C) \subseteq \text{x}(P)$ and $\text{int}(C) \subseteq \text{x}(Q)$ and $\text{ext}(C) = \varnothing$

$\Rightarrow |\text{x}(P) \cap \text{x}(Q)| > 1$ and $\neg(\exists\ R_1, \ldots R_n \in O$: $\forall\ i \in \{1,\ldots,n\}$: $R_i \leftarrow P,Q$ and $\text{x}(R_1) \cup \ldots \cup \text{x}(R_n) = \text{x}(P) \cap \text{x}(Q))$

$\Rightarrow$ violation of contract rule

QED

Table 3. Expanded formal context

	enter	leave	acquire	classify	borrow	renew	return	sell	lose	reserve	cancel
Member	X	X			X	X	X		X	X	X
Book			X	X	X	X	X	X	X	X	X
Loan					X	X	X		X		

Table 4. Formal context with object type Reservation

	enter	leave	acquire	classify	borrow	renew	return	sell	lose	reserve	cancel
Member	X	X			X	X	X		X	X	X
Book			X	X	X	X	X	X	X	X	X
Loan					X	X	X		X		
Reservation										X	X

We now illustrate Theorem 6. Suppose that besides borrowing books, it is also possible to reserve books that are not on shelf. If a member changes her mind and decides not to fetch the copy, she can cancel the reservation. The events "reserve" and "cancel" are added to the object-event table from Table 1. The resulting object-event table is displayed in Table 3.

The shaded area of Table 3 shows the common event types of Book and Member. Some of the events are also in the alphabet of the dependent object type Loan but "reserve" and "cancel" do not appear in the alphabet of a common existence dependent object type. Figure 3 displays the corresponding lattice when the object type Reservation is missing.

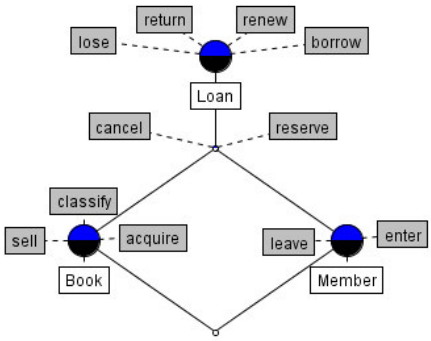

Fig. 3. Potential missing object type

The concept with own attributes "reserve" and "cancel" does not have any own object types in its extent. To fulfill the requirements imposed by the MERODE contract rule, an object type that participates in the own event types contained in the intent of the concept should be added to the business domain model. This object type must be existence dependent of the object types contained in the extent of the concepts one level lower in the lattice.

In this case, according to the MERODE contract rule, the two event types should either be included in the alphabet of Loan or they should be included in the alphabet of a new object type Reservation, dependent of both Member and Copy. According to MERODE-practices, the latter solution is to be preferred, because a loan can occur without a reservation and a reservation can occur without being followed by a loan. Figure 4 displays the correct object-event lattice.

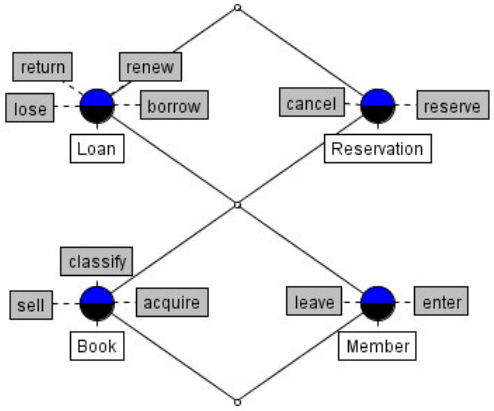

Fig. 4. Correct object-event lattice

However, FCA does not provide any formal grounding for the contract rule and in particular for the requirement that the contract rule should only be applicable in case of two or more common event types. One could for example already consider to create an additional object type even if there is a node with only one event type and no own object type. Also, a node with a potential missing object type as in Fig.3, has already a non-empty extension. So, there is no immediate reason to add an object in that point of the lattice. As a result, FCA does not offer a formal foundation for this rule in MERODE. In fact, in MERODE, this rule was created for deadlock verification purposes [26], rather than to actually identify missing object types. The fact that FCA does not immediately support this rule, could motivate a revision of this rule.

5.4 Verification process

The results of this paper can be used for formal verification purposes as part of a larger software engineering process. The verification process using FCA can be seen as an iterative learning loop. In each iteration, an existing process-data matrix is used to automatically derive an FCA lattice. As we have shown in the previous sections, this lattice can be considered as a conceptual domain model that obeys the wellformedness rules as imposed by MERODE. The lattice structure can then be used for formal verification purposes, i.e. to detect anomalies, missing concepts, missing object types, etc. These changes can be implemented in the existing

process-data matrix and a new iteration through the learning loop is started until a correct model is obtained.

6. Conclusions

In this paper, we proposed a novel application of FCA, namely as a formal foundation for the verification of matrices used in conceptual domain modeling. The well-formedness rules for the object-event table in MERODE were developed by reasoning on the semantics of existence dependency, on common sense reasoning and on process algebra considerations. We showed that by imposing the well-formedness rules from MERODE, we obtain a domain model that has all the properties of a concept lattice. This substantiates the well-foundedness of these rules. Reversely, if FCA is used to cluster an object-event matrix, the concepts identified by FCA can easily be mapped to object types in an enterprise model. In addition, the subconcept-superconcept relationship between FCA-concepts can be mapped to the existence dependency relationship in the enterprise model. And finally, we showcased that one of the rules that was created purely for deadlock verification purposes, can indeed not be grounded in the theory of Formal Concept Analysis. In this way, we demonstrated the applicability of FCA as a theory to aid in the development of sound conceptual modeling methods.

The theory of FCA can also be applied to the is-a relationship between concepts. Future research will investigate the application of FCA to generalisation/specialisation lattices and how this can be combined with a CRUD-like matrix. Yet we already dare to postulate as a general conclusion that FCA is a valid instrument for formalizing the construction of a conceptual domain model. In the future, the work presented in this paper will be empirically validated by applying it to a real life case study using data from a banking company.

Acknowledgements

Jonas Poelmans is Aspirant of the "Fonds voor Wetenschappelijk Onderzoek – Vlaanderen" (FWO) or Research Foundation – Flanders.

.

References

[1] Wand, Y., Weber, R.A., Research commentary: information systems and conceptual modeling - a research agenda, *Information Systems Research,* 13 (4), 2002, 363–376.

[2] Jackson, M.A. *System development*. Prentice Hall, 1983 Englewood Cliffs, New Jersey.

[3] Paige, R., Ostroff, J., The Single Model Principle, *Journal of Object Technology,* vol (5), 2002, 63–81.

[4] Moody, D.L., Theoretical and practical issues in evaluating the quality of conceptual models: current state and future directions. *Data & Knowledge Engineering*, 55, 2005 243-276.

[5] Poels, G., Nelson, J., Genero, Marcela, Piattini, M. (2003) Quality in Conceptual Modeling new research directions. A. Olivé (Eds): *ER* 2003 Ws, LNCS 2784, 243-250.

[6] Lonsdale Systems, Matrix Analysis, available from http://www.lonsdalesystems.com/dokuwiki/doku.php?id=training:requirements_analysis:requirements_analysis, accessed 02/2009.

[7] Stumme, G., Formal Concept Analysis on its Way from Mathematics to computer science. *Proc. 10th Intl. Conf. On Conceptual Structures* (ICCS 2002), LNCS, Springer, Heidelberg.

[8] Ganter, B., Wille, R., *Formal Concept Analysis: Mathematical foundations.* Springer, Heidelberg, 1999.

[9] Wille, R. Restructuring lattice theory: an approach based on hierarchies of concepts. I. Rival (ed.). *Ordered Sets*. Reidel. Dordrecht-Boston, 1982, 445-470.

[10] Snelting, G. Software reengineering based on concept lattices. *Proc. IEEE 4th European Conference on Software Maintenance and Reengineering*, 2000, 3-12,

[11] Snelting, G., Tip, F., Reengineering class hierarchies using concept analysis. *Proc. of ACMSIGSOFT Symposium on the Foundations of Software Engineering*, 1998, 99-110.

[12] Tonella, P., Antoniol, G. Object-oriented design pattern inference. *Proc. of CSM*, 1999, 230-240.

[13] Düwel, S. (1999) Enhancing system analysis by means of formal concept analysis. *Proc. conference of Advanced information systems engineering 6th doctoral consortium*, Heidelberg, Germany, 1999.

[14] Düwel, S., Hesse, W. Identifying candidate objects during system analysis. *Proc. CAiSE'98/IFIP 8.1 3rd Int. Workshop on Evaluation of Modeling Methods in System Analysis and Design* (EMMSAD), 1998.

[15] Richards, D., Boettger, K. A controlled language to assist conversion of use case descriptions into concept lattices. *Proc. of 15th Australian joint conference on artificial intelligence*, LNAI, vol. 2557, 2002, 1-11.

[16] Richards, D., Boettger, K., Fure, A., Using RECOCASE to compare use cases from multiple viewpoints. *Proc. of the 13th Australian Conference on Information Systems* ACIS 2002, Melbourne.

[17] Cimiano, P., Hotho, A., Stumme, G., Tane, J. Conceptual Knowledge Processing with Formal Concept Analysis and Ontologies. *Proc. of ICFA*, LNAI 2961, 2004, 189-207.

[18] Bain, M. Inductive Construction of Ontologies from Formal Concept Analysis. *Proc. of AI*, LNAI 2903, 2003, 88-99.

[19] Godin, R., Mili, H., Mineau, G.W., Missaoui, R., Arfi, A., Chau, T.-T., Design of class hierarchies based on concept (Galois) lattices. *Theory and Application of Object Systems* (RAPOS), 4(2), 1998, 117-134.

[20] Martin, J., *Information Engineering*. Prentice Hall, 1990, Englewood Cliffs, New Jersey.

[21] Robinson, K., Berrisford, G., *Object-Oriented SSADM.* Prentice Hall, 1994, Englewood Cliffs, New jersey.

[22] Snoeck, M., Dedene, G., Verhelst, M., Depuydt, A.M. *Object-Oriented Enterprise Modeling with MERODE*. Leuven University press, 1999.

[23] Snoeck, M., Dedene, G. Existence dependency. The key to semantic integrity between structural and behavioral aspects of objects types. *IEEE Transactions on Software Engineering*, 24(4), 1998, 233-251.

[24] OMG, Unified Modeling Language, Available from www.omg.org/uml, accessed 10/2008.

[25] Chen, P.P. The entity relationship approach to logical database design. *QED information sciences,* Wellesley (Mass.), 1977.

[26] Dedene G., Snoeck M., Formal deadlock elimination, *Data & Knowledge Engineering*, 15(1), 1995, 1-30.