# Relational Contexts and Conceptual Model Clustering

Giancarlo Guizzardi[1], Tiago Prince Sales[1], João Paulo A. Almeida[2], and Geert Poels[3]

[1] Conceptual and Cognitive Modeling Research Group (CORE),
Free University of Bozen-Bolzano, Bolzano, Italy
{giancarlo.guizzardi, tiago.princesales}@unibz.it
[2] Ontology & Conceptual Modeling Research Group (NEMO),
Federal University of Espírito Santo, Vitória, Brazil
jpalmeida@ieee.org
[3] UGent Business Informatics Research Group,
Ghent University, Ghent, Belgium
geert.poels@ugent.be

**Abstract.** In recent years, there has been a growing interest in the use of reference conceptual models to capture information about complex and sensitive business domains (e.g., finance, healthcare, space). These models play a fundamental role in different types of critical semantic interoperability tasks. Therefore, it is essential that domain experts are able to understand and reason with their content. In other words, it is important for these reference conceptual models to be *cognitively tractable*. This paper contributes to this goal by proposing a model clustering technique that leverages the rich semantics of ontology-driven conceptual models (ODCM). In particular, the technique employs the notion of *Relational Context* to guide automated model breakdown. Such Relational Contexts capture all the information needed for understanding entities "qua players of roles" in the scope of an objectified (reified) relationship (relator).

**Keywords:** Conceptual Model Clustering, Complexity Management in Conceptual Modeling

## 1 Introduction

In recent years, there has been a growth in the use of reference conceptual models, in general, and domain ontologies, in particular, to capture information about complex and critical domains [11]. However, as the complexity of these domains grows, often so does the sheer size and complexity of the artifacts that represent them. Moreover, in sensitive domains (e.g., finance, healthcare), these models play a fundamental role in different types of critical semantic interoperability tasks, therefore, it is essential that domain experts are able to understand and accurately reason with the content of these models. The human capacity for processing unknown information is very limited, containing bottlenecks in visual short-term memory and causing problems to identify and hold stimuli [18]. Therefore, there is an evident need for developing adequate complexity management mechanisms for reference conceptual models.

One type of such complexity management mechanisms is conceptual model modularization or *Conceptual Model Clustering* (henceforth CMC) [1]. CMC is the process

by which a model is fragmented into smaller interconnected parts [17], each of which can be more easily manipulated by a model user than the entire model. The greatest challenge in CMC is the process for *module extraction*, namely, coming up with adequate criteria for dividing the model into modules that ease model understanding.

Traditionally, different techniques have been used for module extraction (e.g., [1, 16]). However, almost the totality of these approaches address modularization in languages that are *ontologically-neutral* [14] such as UML, ER diagrams or OWL[4] .While these languages may have a well-defined abstract syntax and a formal (logical) semantics, in general, they lack an *ontological semantics*. Consequently, the modularization techniques developed for them rely on criteria that leverage almost exclusively on the syntactical properties of the models, typically, topological ones [28].

In contrast, ontology-driven conceptual modeling (ODCM) languages are systematically designed to conform to an underlying ontological theory. In particular, an ODCM language contains exactly the modeling primitives that are necessary to represent the ontological distinctions put forth by its underlying ontology. ODCM approaches have enjoyed an increasing adoption by the Conceptual Modeling community as a number of independent results consistently show their benefits for improving the quality of conceptual models (e.g., [27]). An example of an ODCM language is OntoUML [10], whose primitives reflect the underlying UFO foundational ontology [10].

In this paper, we leverage the ontologically well-founded semantics of OntoUML to propose a formal approach for automated modularization in conceptual models. The proposed approach breaks down an OntoUML model in a number of *Relational Contexts*. Intuitively, Relational Contexts are modules that capture all the information needed for understanding entities *qua players of roles* in the scope of an objectified (reified) relationship (ontological speaking, the so-called *relators*).

As reported in [23], *Relators* and *Roles* are clearly the most used OntoUML constructs in conceptual models. This is unsurprising, given the strong adoption of OntoUML/UFO in Business (Organizational/Social/Legal) domains [26], as well as the fact that in these realms the bulk of the domain knowledge is concentrated in relationships and roles. As argued in [12], specially in these realms, "we seldom interact with these entities *qua-themselves*, but we frequently conceive objects *qua-playing-certain-roles* in given 'contexts'... For example, most of our interactions with other human beings and, hence, our conceptualizations of these interactions are thought in terms of roles such as parent, employee, student, president, citizen, customer, etc. Analogously, when thinking about, for instance, cars, we think about them as means of transportation, insurable items, work-related resources, product offerings, etc. Moreover, we often conceive these 'contexts' as *relational* ones: marriages, employments, enrollments, and presidential mandates are themselves concrete 'object-like' entities that define a scope in which ordinary objects play complementary roles interacting with each other". This view is also defended by other authors such as [4], who go as far as to claim that "[r]oles are useful not only to model domains that include institutions and organizations. Rather, every object can be considered as an institution or an organization structured in roles".

[4] There is a long debate in philosophy regarding the ontological neutrality (or lack thereof) of formal languages. We simply mean here that they commit to a simple ontology of formal structures (e.g., that of set theory) in which sorts of types and relations are undifferentiated.

The proposal advanced here is, thus, aimed at conceptual models in business (organizational, social and legal) domains, which form the bulk of the Information Systems discipline. For models that are centered on taxonomic relations (e.g., product types, biological taxonomies), we recommend alternative complexity management techniques, in particular, the static *ontological views* as proposed in [6]. In fact, this paper can be seen as a companion to [6] and [13] in a general research program of defining ontology-driven complexity management theories, techniques and tools. While in these two papers the focus is on *model recoding with ontology-design patterns*, and on *model abstraction*, respectively, here we propose the notion of relationship-centric conceptual model modularization (or clustering).

The contributions of this paper are two-fold: (i) firstly, we proposed a formalization of the notion of *Relational Context* by leveraging on the theory of relators from UFO/OntoUML; We then use this notion to propose a strategy for relationship-centric modularization termed *Relator-Centric Clustering*; (ii) secondly, we provide an implementation of this strategy integrated in the OntoUML toolset.

The remainder of the paper is organized as follows. Section 2 positions our work in reference to related efforts; Section 3 briefly presents the OntoUML language and some of the ontological notions underlying it; Section 4 presents the contributions of this paper. Firstly, it defines the notions of Ontological Views, Relational Contexts, and Modular Breakdown. This is done both formally, in terms of a precise definition of these notions, as well as intuitively by making use of a running example in the domain of Car Rental. In addition, we report on an implementation of this approach as a plug-in to a model-based OntoUML editor. finally, Section 5 presents some conclusions of the presented approach and some intended directions for future work.

## 2 Complexity Management of Conceptual Models

The discipline of *complexity management of large conceptual models* (henceforth CM-CM) has been around for quite some time and has been represented in the literature by a series of different approaches and techniques. In fact, [28] claims that "one of the most challenging and long-standing goals in conceptual modeling... is to understand, comprehend and work with very large conceptual schemas".

The challenge and importance of this discipline lie in the following. On one hand, real information systems often have large and extremely complex conceptual models [28]. On the other hand, this complexity poses a serious additional challenge in the comprehension and, consequent, quality assurance of these models. For example, [21] reports on an empirical study conducted with a large and professionally constructed conceptual model.[5] In that study, the authors managed to show that the model contained 879 occurrences of error-prone structures (anti-patterns), 52.56% of which really introduced representation errors according to the creators of the model.

According to [28], the methods for CM-CM can be classified in three areas, namely, *Clustering Methods*, *Relevance Methods*, and *Summarization Methods*. Clustering is about classifying the elements of a conceptual model into groups, or clusters, according

[5] This model consisted of 3,800 classes, 61 datatypes, 1,918 associations, 3,616 subtyping relations, 698 generalization sets, 865 attributes, i.e., navigable association ends [21].

to some criteria (e.g., a similarity function); Relevance Methods are about the application of ranking functions to the elements of a model in order to obtain ordered lists (i.e., a ranking) of model elements according to their perceived relevance in representing the domain at hand; finally, Model Summarization is about producing from an original model a reduced version consisting only of the elements that are judged to be of more relevance for representing the domain at hand. In clustering methods, the goal is to break down a model in fragments such that the sum of these fragments should be informationally equivalent to the whole (i.e., to the original model). In contrast, relevance and summarization methods (including *model abstraction*) aim to produce partial views of the original model at hand. In other words, while clustering methods have *lossless model transformations*, the latter classes of methods are based on *lossy transformations*.

A drawback that is common to the majority of existing methods in all these classes is that they are based on classic conceptual modeling notations (e.g., UML, ER) [28], they are constrained to rely almost exclusively on syntactic (mainly topological) properties of the addressed models. These properties include *closeness* (a quantitative evaluation of the links among elements in the model) [8], *hierarchical distance* (length of the shortest relationship path between entities), *structural-connective distance* (elements are considered closer if they are neighbors in a hierarchy mereological or subtyping structure), or *category distance* (elements are considered to be closer if one subtypes the other) [1]. For example, [5] proposes a (relevance) method based on the assumption that the number of attributes and relations characterizing an element in a model can be used as a (heuristic) measure of its relevance for that model. In the same spirit, [24, 25] go as far as proposing PageRank-style algorithms to infer the relevance of elements in entity-relationship diagrams and RDF schemas (even ignoring the difference between association and subtyping relations). The problem with relying solely on these properties is that there is no guarantee that a model element satisfying some topological requirement (e.g., a node with more edges connected to it) by necessity represents the model's most important concepts. This is related to the work by [19, 20], that while criticizing existing CM-CM methods, referred to it as *lack of cognitive justification*.

The method proposed here is a type of clustering method. However, in contrast with all the aforementioned approaches, our proposal focuses mainly on the *ontological semantics* [10] of the elements represented in a conceptual model. As previously discussed, the idea is to use a formal and ontological notion of *Relational Context* (see section 4) as a clustering mechanism. Relational Contexts are built from a focal reified relationship (*relator*), and extrapolating from there on to the different *roles* played by entities in the scope of that relationship, the *kinds* defining the essential properties and identity principle characterizing the entities playing these roles, among other aspects.

This approach (detailed in section 4) is only made possible because it is based on a non-classical CM language, namely, the ODCM language OntoUML (briefly presented in section 3). There are three CM-CM methods in the literature that are based on the same language, namely, the approaches of (i) [6], (ii) [13], and (iii) [16, 17]. The first method is the one that is closer to work presented here, since it is also a clustering method and, hence, a *lossless approach*. What is presented there is an approach for what the authors name *Model Recoding*. The method takes a conceptual model and produces a series of views constituted by ontological design patterns centered around

general (as opposed to model specific) ontological constructs. So, for example, it groups all the kinds of things the model in one view, all the roles played by things in a relational context in another view, etc. So, instead of breaking down the model into clusters that correspond to what one could intuitively call *sub-domains*, that approach brakes down the model in terms of general ontological categories. In contrast, the approaches of [13] and [16, 17] differ from the approach presented here since these are approaches for model summarization and hence *lossy approaches*. Finally, [16, 17] also differs from our approach since it requires user input in selecting a set of entities in the model that are of particular relevance. Our approach, instead, is a fully automated one, which we argue is an important feature in methods dealing with large-scale models.

## 3 A Whirlwind Introduction to UFO and OntoUML

OntoUML is a language whose meta-model has been designed to comply with the ontological distinctions and axiomatization of a theoretically well-grounded foundational ontology named UFO (Unified Foundational Ontology) [10, 15]. UFO is an axiomatic formal theory based on contributions from Formal Ontology in Philosophy, Philosophical Logic, Cognitive Psychology, and Linguistics. A recent study shows that UFO is the second-most used foundational ontology in conceptual modeling and the one with the fastest adoption rate [26]. That study also shows that OntoUML is among the most used languages in ontology-driven conceptual modeling.

In the sequel, we briefly explain a selected subset of the ontological distinctions put forth by the Unified Foundational Ontology (UFO). We also show how these distinctions are represented by the modeling primitives of OntoUML (as a UML profile). For an in-depth discussion, philosophical justifications, formal characterization and empirical support for these categories one should refer to [9, 10].

Take a domain in reality restricted to endurants [10] (as opposed to events or occurrents). Central to this domain we will have a number of object *Kinds*, i.e., the genuine fundamental types of objects that exist in this domain. The term "kind" is meant here in a strong technical sense, i.e., by a kind, we mean a type capturing essential properties of the things it classifies. In other words, the objects classified by that kind could not possibly exist without being of that specific kind.

Kinds tessellate the possible space of objects in that domain, i.e., all objects belong to exactly one kind and do so necessarily. Typical examples of kinds include Person, Organization, and Car (see Figure 1; stereotypes reflect the correspondence between the UML profile and UFO[6]). We can, however, have other static subdivisions (or subtypes) of a kind. These are naturally termed *Subkinds*. As an example, the kind 'Person' can be specialized in the subkinds 'Man' and 'Woman' (Figure 1).

[6] The model of Figure 1 is used here for illustration purposes only, as it is a much simplified version of a proper model in this domain. For example, in a more realistic model, we would have cases of "relators mediating relators" (e.g., a car rental mediating a car ownership and an employment). The example avoids these for the sake of space limitations. Our formal definition of RCC (see section 4.7), however, has no such a limitation, thus, addressing these cases that result in *nested contexts* (i.e., contexts including other contexts).

Object kinds and subkinds represent essential properties of objects (they are also termed rigid or static types [10]). We have, however, types that represent contingent or accidental properties of objects (termed anti-rigid types [10]). These include *Phases* (for example, in the way that 'being a living person' captures a cluster of contingent *intrinsic* properties of a person, or in the way that 'being a puppy' captures a cluster of contingent *intrinsic* properties of a dog) and *Roles* (for example, in the way that 'being a husband' captures a cluster of contingent *relational* properties of a man participating in a marriage, or that 'being a rental car' captures contingent *intrinsic* properties of a car participating in a car rental, see Figure 1). In other words, the difference between the contingent properties represented by a phase and a role is the following: phases represent properties that are intrinsic to entities (e.g., 'being a puppy' is being a dog that is in a particular developmental phase; 'being a living person' is being a person who has the intrinsic property of being alive; 'being an available car' is being a car that is functional and, hence, can be rented); roles, in contrast, represent properties that entities have in a relational context, i.e., contingent relational properties (e.g., 'being a husband' is to bear a number of commitments and claims towards a spouse in the scope of a marital relationship; 'being a student' is to bear a number of properties in the scope of an enrollment relationship with an educational institution.)

Kinds, Subkinds, Phases, and Roles are categories of object *Sortals*. In the philosophical literature, a sortal is a type that provides a uniform principle of identity, persistence, and individuation for its instances [10]. To put it simply, a sortal is either a kind (e.g., 'Person') or a specialization of a kind (e.g., 'Student', 'Teenager', 'Woman'), i.e., it is either a type representing the essence of what things are or a sub-classification applied to the entities that "have that same type of essence".

*Relators* (or relationships in a particular technical sense [9]) represent clusters of relational properties that "hang together" by a nexus (provided by a relator kind). Moreover, relators (e.g., marriages, enrollments, presidential mandates, citizenships, but also car rentals, employments, and car ownerships, see Figure 1) are full-fledged *Endurants*. In other words, entities that endure in time bearing their own essential and accidental properties and, hence, first-class entities that can change in a qualitative manner while maintaining their identity.

As discussed in depth in [9], relators are the truth-makers of relational propositions, and relations (as classes of n-tuples) can be completely derived from relators [10]. For instance, it is 'the marriage' (as a complex relator composed of mutual commitments and claims) between 'John' and 'Mary' that makes true the proposition that "John is the husband of Mary". Relators are existentially dependent entities (e.g., the marriage between John and Mary can only exist if John and Mary exist) that bind together entities (their relata) by the so-called *mediation* relations - a particular type of existential dependence relation [10]. As discussed in depth in [9], like in the MERODE approach [22] (but here for ontological reasons), all domain relations in business models (the so-called material relations) can be represented exclusively by employing relators and these existential dependence relations (mediation).

Objects participate in relationships (relators) playing certain "roles". For instance, people play the role of spouse in a marriage relationship; a person plays the role of president in a presidential mandate; a car plays the role of a rental car scope of a car

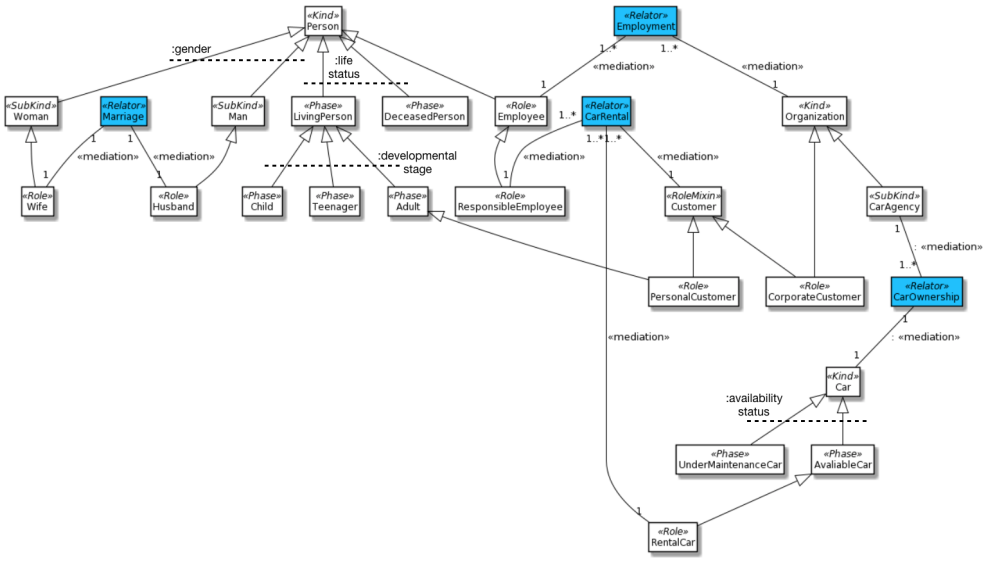


**Fig. 1.** A conceptual model in OntoUML in which relators are highlighted.

rental, see Figure 1. 'Spouse' and 'President' (but also typically student, teacher, pet) are examples of what we technically term a role in UFO, i.e., a relational contingent sortal (since these roles can only be played by entities of a unique given kind). There are, however, relational and contingent role-like types that can be played by entities of multiple kinds. An example is the role 'Customer' (which can be played by both people and organizations), see Figure 1. We call these role-like types that classify entities of multiple kinds *RoleMixins*.

## 4 Views, Relational Contexts, and Relator-Centric Clustering

In this section, we present a formal definition of our structure of ontological views, which are then used to formally define our notion of Relational Context (RC) and of Relator-Centric Clustering (RCC). Built over UFO's distinctions and for the OntoUML language, the approach presented here proposes rules to extract modules (clusters) from a conceptual model expressed in OntoUML.

### 4.1 Basic Definitions

Let a Model $M$ be a graph defined such that $M = \langle \Theta, \Sigma, \Phi \rangle$ , where $\Theta = \{C_1..C_n\}$ is the (non-empty) set of concepts in the model $M$), $\Sigma = \{r_1..r_n\}$ is the set of directed relations in the model and $\Phi = \{gs_1..gs_n\}$ is the set of Generalization Sets in the model. Let $CT$ (Concept Type), $RT$ (Relation Type) and $GST$ be domains of types such that $CT$ = {SORTAL, NON-SORTAL, KIND, SUBKIND, PHASE, ROLE,

ROLEMIXIN, RELATOR}, $RT = \{$MEDIATION, SUBTYPING$\}$, and $GST = \{$PHASE-PARTITION, SUBKIND-PARTITION$\}$ . Now, let $<$ be partial order relation defined in $CT$ in the following way to reflect the specializations in the taxonomy of types in UFO: KIND $<$ SORTAL, SUBKIND $<$ SORTAL, ROLE $<$ SORTAL, PHASE $<$ SORTAL, ROLEMIXIN $<$ NON-SORTAL. Finally, we define a number of auxiliary functions:

- $C(M)$ is a function that maps a model $M$ to its associated set $\Theta$;
- $R(M)$ is a function that maps a model $M$ to its associated set $\Sigma$;
- $GS(M)$ is a function that maps a model $M$ to its associated set $\Phi$;
- $E\ HasType\ T$ is a relation connecting an element $E$ to a type $T$ in the following manner: if $E$ is a concept, then $T \in CT$; if $E$ is a relation then $T \in RT$, and if $E$ is a generalization set, then $E \in GST$. We should also add that for any two types $T$ and $T'$ such that $T < T'$, if $E\ HasType\ T$ then $E\ HasType\ T'$;
- $t(r)$ is a function that maps a relation $r$ to the target (destination) of that directed relation;
- $s(r)$ is the complementary function that maps a relation $r$ to the source (origin) of that directed relation;
- $r \triangleright gs$ connects a relation $r$ with a generalization set $gs$ such that $r\ HasType$ SUBTYPING and: if $gs\ HasType$ PHASE-PARTITION then $s(r)\ HasType$ PHASE; if $gs\ HasType$ SUBKIND-PARTITION then $s(r)\ HasType$ SUBKIND. Moreover, for any two relations $r_1$ and $r_2$ such that $r_1 \triangleright gs$ and $r_2 \triangleright gs$, we have that $t(r_1) = t(r_2)$.

As expected, we have that for every model $M$ and every relation such that $r \in R(M)$, we have that both $s(r) \in C(M)$ and $t(r) \in C(M)$. Moreover, every generalization set $gs \in GS(M)$ is such that all $r \triangleright gs$ implies that $r \in R(M)$.

For example, let $M$ be the model depicted in Figure 1. Then, $C(M)$ amounts to exactly the types represented there, while $R(M)$ includes all the mediation and UML subtyping relations. Finally, $GS(M)$ amounts to the generalization sets: *gender* (a subkind partition comprising the subtyping relations connecting *Man* to *Person*, and *Woman* to *Person*); *developmental status* (a phase partition comprising the subtyping relations connecting *Child* to *Person*, *Teenager* to *Person*, and *Adult* to *Person*); *Life Status* (a phase partition comprising subtyping relations connecting *Living Person* to *Person*, and *Deceased Person* to *Person*); *Operational Status* (a phase partition comprising subtyping relations connecting *Available Car* to *Car*, and *Under Maintenance Car* to *Car*).

### 4.2 Direct Subtyping and (Indirect) Subtyping

Let the functions $ST(C,C')$ (symbolizing that $C$ is a direct subtype of $C'$), $ST*(C,C')$ (symbolizing that $C$ is a subtype of $C'$) and $IST*(C,C')$ (symbolizing that $C$ is an improper subtype of $C'$) be defined as follows:

- $ST(C,C')$ iff there is an $r$ such that $r\ HasType$ SUBTYPING and $s(r) = C$ and $t(r) = C'$;
- $ST*(C,C')$ iff $ST(C,C')$ or there is a $C''$ such that $ST(C,C'')$ and $ST*(C'',C')$; and,
- $IST*(C,C')$ iff $ST*(C,C')$ or $C = C'$.

We also define the following auxiliary function:

– $K(C)$ mapping a sortal $C$ to its unique supertyping KIND, i.e., we have that $K(C) = C'$ iff $C'$ *HasType* KIND and $IST*(C,C')$. (Notice that if $C$ is a KIND, then $C = C'$.)

Again, using the model $M$ of Figure 1 as an example, we have that, for instance, $K(CarAgency) = Organization$ and $K(PersonalCustomer) = Person$.

### 4.3 View

Let $M$ and $M'$ be models as previously defined. It follows that $M$ is a view of $M'$ (symbolized as $V(M,M')$ iff:

– $C(M) \subseteq C(M')$ and
– $R(M) \subseteq R(M')$ and
– $GS(M) \subseteq GS(M')$.

Notice that, given our definition of a model, we have that all $r \in R(M)$ are such that $s(r) \in C(M)$ and $t(r) \in C(M)$, but also that for all $r \triangleright GS(M)$ we have that $r \in R(M)$. In other words, $M$ is necessarily an original subgraph of $M'$.

The views we are ultimately interested in are the so-called Relational Contexts (RC), which will be defined in sub-section 4.6. Nevertheless, before we reach that, we need to establish two types of auxiliary views: Sortal Identity Paths and Non-Sortal Identity Paths. They are used later to support the definition of Relational Contexts.

### 4.4 Sortal Identity Path

We define that a view $M$ is a Sortal Identity Path of $M'$ based on a focus type $c$ (symbolized as $SIP(M,M',c)$, where $c$ *HasType* SORTAL iff:

– $V(M,M')$ and
– $c' \in C(M)$ iff ($IST*(c,c')$ and $IST*(c',K(c))$ and
– $r \in R(M)$ iff $r$ *HasType* SUBTYPING and $s(r) \in C(M)$ and $t(r) \in C(M)$.

SIP is a generic parameterizable view definition that, given a sortal type $c$, it provides with a view that includes that type and all its supertypes (if any) until its corresponding kind is reached. Taking the model of Figure 1 and picking, for instance, *Personal Customer* as focus type, the corresponding SIP would be constituted by the types that generalize *Personal Customer*, i.e., *Adult*, *Living Person*, and, finally, *Person*. Later, we use SIP to determine which supertypes should be included in a Relational Context, namely those that reveal the nature of the entities in the context.

### 4.5 Non-Sortal Identity Paths

We define that the view $M$ is a Non-Sortal Identity Paths of $M'$ based on a focus type $c$ (symbolized as $NSIP(M,M',c)$, where $c$ *HasType* NON-SORTAL iff:

– $V(M,M')$ and

- $c' \in C(M)$ iff $IST*(c', c)$ or (there is a $c''$ such that $IST*(c'', c)$ and $IST*(c'', c')$ and $IST*(c', K(c'')))$ and
- $r \in R(M)$ iff $r$ *HasType* SUBTYPING and $(s(r) \in C(M))$ and $(t(r) \in C(M)))$.

The intention of the *NSIP* can explained as follows. Take a non-sortal type $c$ in the model $M'$, this view should include: (i) $c$ itself and all its non-sortal subtypes; (ii) the first sortal specializing $c$ as well as the path from this sortal to the unique kind providing its *identity principle* [10]. Taking the model of Figure 1 and picking, for instance, *Customer* as focus type, in the corresponding *NSIP*, we have, besides the rolemixin *Customer*, the sortals that immediately specialize it (the roles *Personal Customer* and *Corporate Customer*) as well as the supertypes of each of these sortals that are in the path between them and their kinds (*Person* and *Organization*, respectively, in this case).

### 4.6 Relational Context

We define that $M$ is a Relational Context of $M'$ with focus on a relator type *rel*, where (*rel HasType* RELATOR) (symbolized as $RC(M, M', rel)$) iff the following conditions are satisfied:

- $V(M, M')$;
- $c \in C(M)$ iff:
  - $c = rel$, or
  - there is a $r \in R(M)$ and $t(r) = c$, or
  - there is a view $M''$ and a $c' \in C(M)$ such that $(SIP(M'', M', c')$ or $NSIP(M'', M', c'))$ and $c \in C(M'')$, or
  - there is a $gs \in GS(M)$ and a $r \triangleright gs$ and $s(r) = c$, or
- $r \in R(M)$ iff:
  - ($r$ *HasType* MEDIATION and $s(r) \in C(M)$) or
  - ($r$ *HasType* SUBTYPING and $s(r) \in C(M)$) and (($t(r)$ *HasType* RELATOR) or $t(r) \in C(M)$), or
  - there is a $gs \in GS(M)$ such that $r \triangleright gs$
- $gs \in GS(M)$ iff:
  - $gs$ *HasType* PHASE-PARTITION and there is an $r$ such that $r \triangleright gs$ and $r \in R(M)$, or
  - $gs$ *HasType* SUBKIND-PARTITION and for all $r$ such that $r \triangleright gs$ then $r \in R(M)$.

Now, this definition can benefit from some unpacking. The Relational Context (RC) starts by (naturally) including the focal relator *rel* $(c = rel)$. In addition, it includes all types that are connected by that relator via MEDIATION relations (henceforth, *mediated types*) (($r \in R(M)$ and $t(r) = c$) and ($r$ *HasType* MEDIATION and $s(r) \in C(M)$)). For example, if we take the relator *Car Rental* as focus, the corresponding RC would also include the types of entities that are bound by instances of *Car Rental* in that context, i.e., *Customer* and *Rental Car*.

Furthermore, this RC should include in this context all the types going from these mediated types to their respective kinds. The rationale here is that in order to understand the nature of the entities connected by instances of the relator at hand, one must

understand what kinds of things those entities *essentially* are, i.e., what sort of *principle of identity* they obey. In case any of these mediated types $c'$ is a sortal, then the RC will include all types in its *SIP* ($c' \in C(M)$ and there is and a view $M''$ such that $SIP(M'', M', c')$ and $c \in C(M'')$). So, in this example, for the sortal type *Rental Car*, it would include also the types *Available Car* and *Car*. In contrast, if any of the mediated types is a Non-Sortal, then the relational context will include all types in its *NSIP* ($c' \in C(M)$ and there is a view $M''$ such that $NSIP(M'', M', c')$ and $c \in C(M'')$). The rationale here is analogous. However, since different instances of a non-sortal might take their identities from different kinds, in order to understand that context, we need to include all the information in the identity path between that non-sortal mediated type and the relevant kinds. For instance, for a *Car Rental* Relational Context, we need to understand the notion of *Customer* and, in order to understand this notion we have to understand the notions of *Personal Customer* and *Corporate Customer*. Finally, in order to understand the latter, we need to understand *Organizations*, and to understand the former, the notions of *Adult*, *Living Person* and *Person*. After all, instances of *Personal Customer* are adult living people.

Besides the types in *SIP* and *NSIP* of mediated types, the Relational Context should also include all types that appear in phase partitions standing in the path between a mediated type and its identity supplier (i.e., its associated kind). The idea is that these types offer a contrast background that helps in the clarification of the semantics of the types in these paths. For example, in the Car Rental context, in order to understand that personal customers must be living adults, it is important to understand that they cannot be other alternatives of instances of *Person*, namely, living children, living teenagers, as well as deceased person. In particular, given the anti-rigidity of these types (phases), all instances of living person can cease to be so, thus, becoming deceased people, in which case they can no longer play the role of *Personal Customer*. Formally, if one of the subtyping relations in a $(N)SIP$ is part of a phase partition, then that phase partition generalization set is included in the view ($gs$ *HasType* PHASE-PARTITION and there is an $r$ such that $r \triangleright gs$ and $r \in R(M)$). Additionally, all other types that share the common supertype in that generalization set are also included in the view (there is a $gs \in GS(M)$ and a $r \triangleright gs$ and $s(r) = c$), and so are all these supertyping relations in that same generalization set ($r$ *HasType* SUBTYPING and $t(r) \in C(M)$ and (there is a $gs$ such that $gs \in GS(M)$ and $r \triangleright gs$)). Notice that subkind partitions are only included (a posteriori) if all subtyping relations comprising it are already included in the view (e.g., *gender* in an RC with *Car Rental* as the focus).

Furthermore, we include in a relational context all subtyping relations involving two types included in that view ($r$ *HasType* SUBTYPING and $s(r) \in C(M)$ and $t(r) \in C(M)$). Finally, we include all supertypes of relators already included in the view ($r$ *HasType* SUBTYPING and $s(r) \in C(M)$ and $t(r)$ *HasType* RELATOR). This is because a subtype inherits all the properties of its supertypes, and thus to understand the context of a sub-relator we must understand the general notion (e.g., to understand 'foreign marriage' as a 'marriage' recognized abroad, we must understand 'marriage' as a relation binding spouses).

### 4.7 Relator-Centric Clustering

We are now in position to define the notion of a *Relator-Centric Clustering*:

- **RCC Definition**: a Relator-Centric Clustering of a model $M$ is a set of views symbolized as $RCC(M) = \{M_1..M_n\}$ such that for every $M_i \in RCC(M)$ there is a type $rel$ such that $rel \in C(M)$ and $RC(M_i, M, rel)$.

Figure 2 depicts the application of this notion of RCC to the model of Figure 1. Here we represent each Relational Context using UML packages and name these packages with the homonymous focal relator. As one can observe, the original model can be broken down into four contexts, namely: the *Car Rental*, the *Marriage*, the *Car Ownership*, and the *Employment* contexts. Each of these modules contains a view of the original model with all the information required to understand each of the contexts.

The *Car Rental* RC shows the roles (and role mixin) directly mediated by the *Car Rental* relator (*Responsible Employee*, *Rental Car*, *Customer*). The kinds involved are made explicit: *Person*, *Car* and *Organization* (when playing the role of *Corporate Customer*). Important business rules the model imposes on a *Car Rental* are revealed: only an *Adult* (a *Living Person*) can rent a car, and only a car that is in the *Available Car* phase can be rented. A similar observation can be made for the *Marriage* RC, as it reveals that the original model reflects a heteronormative setting and with *gender* in static classification. Finally, the *Car Ownership* and the *Employment* RCs are examples of simpler views, as the path from directly mediated entities to the involved kinds is short.

We implemented this approach for relational context identification and relator-centric clustering in Javascript as a service within `ontouml-js`[7], an open source library we have been developing for OntoUML. Currently, this library supports programatically manipulating OntoUML models, automatically verifying their syntax, and automatically transforming them into OWL specifications compatible with gUFO (the reference implementation of the Unified Foundational Ontology in OWL [3]). These services are then made available to final users via the OntoUML plugin[8] for Visual Paradigm.

## 5 Final Considerations

In this paper, we propose a formal approach for conceptual model clustering by leveraging on the ontologically well-founded semantics of the modeling language OntoUML. In particular, we rely on the theory of relators underlying OntoUML to present a full formal account of the notions of *Relational Context* and *Relator-Centric Clustering*. An RCC is a model modular breakdown in terms of a number of adequate RCs. Each RC, in turn, captures all the information needed to understand the maximal scope of objects in the way they participate in certain relationships. The approach is formally characterized (*claim to formal precision*) and it is based on a well-founded ontological theory of relators (*claim to ontological adequacy*).

Additionally, we have reported on a fully implemented plug-in tool for a Model-Based OntoUML Editor that automates this approach (*claim to practical realizability*).

[7] See source code at `https://github.com/OntoUML/ontouml-js`.

[8] See source code at `https://github.com/OntoUML/ontouml-vp-plugin`.

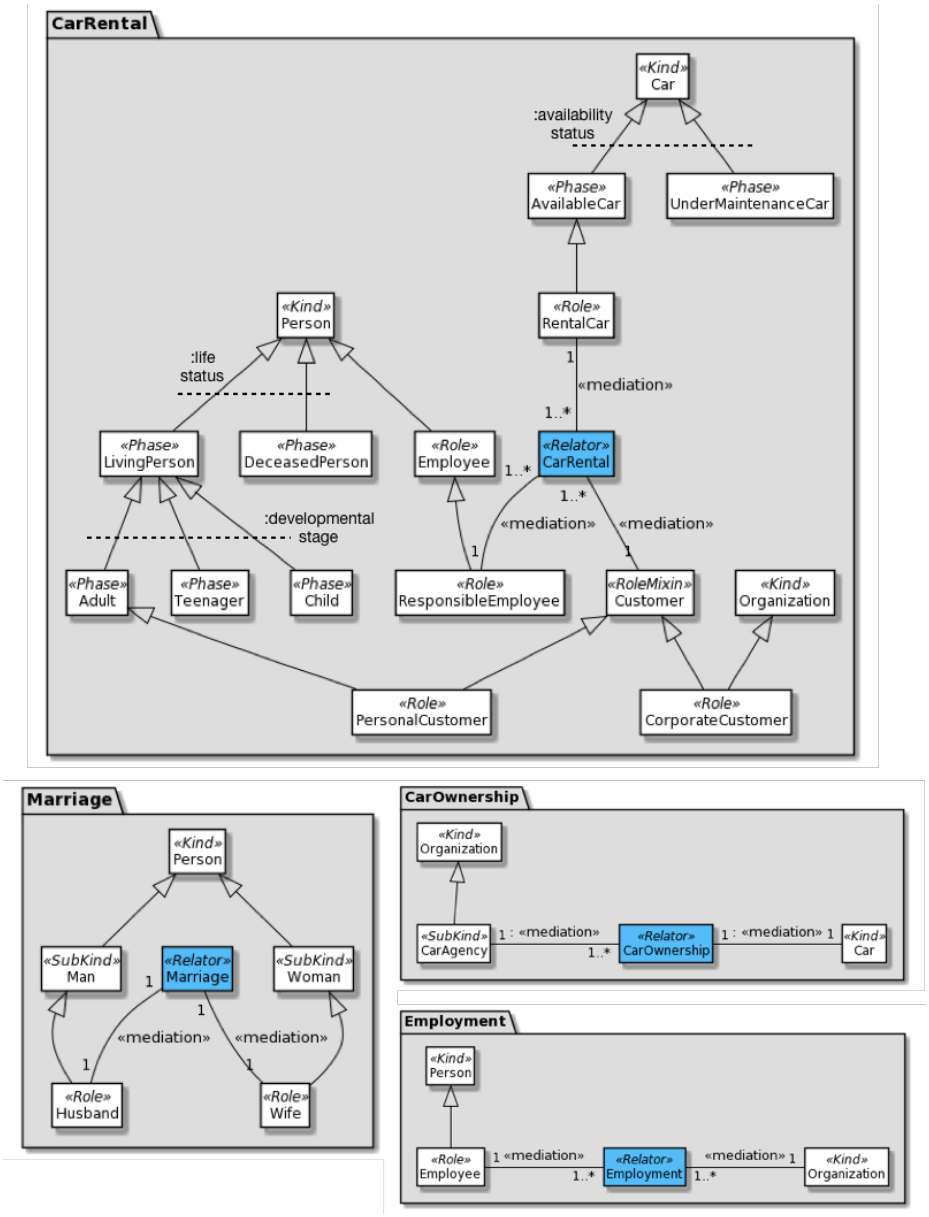


**Fig. 2.** An RCC for the model of Figure 1 organized as (Onto)UML packages.

Following the formal characterization of this framework, the algorithm implementing it is a deterministic one (i.e., it generate the same RCC for a given model in every execution) and, in the worst possible case, the algorithm would execute a total of $(n_e - n_r) * n_r$ operations (where $n_e$ is the total number of model elements in the model and $n_r$ is the total number of classes stereotyped as relators). So, even in the worst possible case, the algorithm is tractable (*claim to computational efficiency and scalability*). In practice, $n_r$ is on average circa 6% of $n_e$ (as observed by analyzing 54 OntoUML models in different domains in the OntoUML repository [21, 23]), and RCs are often largely disjoint with minimal intersections only in the level of kinds. In other words, in practice, the algorithm will often execute approximately $n_e$ steps as the different RCs tessellate the original model. Despite these encouraging results, we intend to subject it to a more comprehensive and systematic analysis and series of tests.

In [2], the authors present an approach for representing reified events (occurrences) as first-class citizens in structural conceptual models. In those models, events have their own properties and can form taxonomic and temporal ordering structures. Moreover, objects participate in these events playing a number of 'processual roles' (e.g., the roles of victim and perpetrator in a crime). As an extension of the approach presented here, we intend to characterize contexts and clusters centered around this notion of events.

The notion of Relational Context proposed here bears a resemblance also to the notion of *Frames* in C.J. Fillmore's Frame Semantics [7]. In fact, we first considered using the term *Ontological Frame* (or *Relational Frame*) for this notion. Frames, in that tradition, are patterns that describe situations, events or relationships and in which elements appear playing interconnected and mutually dependent (semantic) roles. However, unlike our approach, frames have the primary goal of providing a background structure for the interpretation of lexical terms. RCs, in contrast, have as primary goal ontological transparency, focusing on connecting the entities playing complementary roles in the scope of bundles of relational properties (relators) to their identity-providing kinds.

Finally, in order to properly evaluate the *cognitive effectiveness* of these contributions, we are already in the process of designing a series of empirical studies. The core focus concerns speed and recall in obtaining information from the business conceptual model, as well as naturalness to domain experts of the resulting breakdowns.

**Acknowledgments**

We are grateful to Ricardo A. Falbo (*in memoriam*) for the spark that led to this investigation. This research is partially funded by the NeXON Project (UNIBZ). J.P. Almeida is funded by CAPES (grant number 23038.028816/2016-41) and CNPq (grants numbers 312123/2017-5 and 407235/2017-5).

## References

1. Akoka, J., Comyn-Wattiau, I.: Entity-relationship and object-oriented model automatic clustering. Data Know Eng 20(2), 87–117 (1996)
2. Almeida, J.P.A., Falbo, R.A., Guizzardi, G.: Events as entities in ontology-driven conceptual modeling. In: Proc.38th ER. vol. 11788, pp. 469–483. Springer (2019)

3. Almeida, J.P.A., Guizzardi, G., Falbo, R.A., Sales, T.P.: gUFO: A lightweight implementation of the Unified Foundational Ontology (UFO), http://purl.org/nemo/doc/gufo
4. Baldoni, M., Boella, G., van der Torre, I.L.: Interaction between objects in powerjava. J. Object Technol. 6(2) (2003)
5. Castano, S., De Antonellis, V., Fugini, M.G., Pernici, B.: Conceptual schema analysis: Techniques and applications. ACM Trans. Database Syst. 23(3), 286–333 (1998)
6. Figueiredo, G. et al.: Breaking into pieces: An ontological approach to conceptual model complexity management. In: Proc. 12th IEEE RCIS. pp. 1–10 (2018)
7. Fillmore, C.J., et al.: Frame semantics. Cognitive linguistics: Basic readings 34 (2006)
8. Francalanci, C., Pernici, B.: Abstraction levels for entity-relationship schemas. In: Proc. 13th ER. pp. 456–473. Springer (1994)
9. Guarino, N., Guizzardi, G.: "We need to discuss the relationship": Revisiting relationships as modeling constructs. In: Proc. 27th CAISE. pp. 279–294 (2015)
10. Guizzardi, G.: Ontological foundations for structural conceptual models. CTIT, Centre for Telematics and Information Technology (2005)
11. Guizzardi, G.: Ontological patterns, anti-patterns and pattern languages for next-generation conceptual modeling. In: Proc. 33rd ER. pp. 13–27 (2014)
12. Guizzardi, G.: Objects and events in context. In: Proc. 11th CONTEXT (2019)
13. Guizzardi, G., Figueiredo, G., Hedblom, M.M., Poels, G.: Ontology-based model abstraction. In: Proc. 13th IEEE RCIS. pp. 1–13 (2019)
14. Guizzardi, G. et al.: The role of foundational ontologies for domain ontology engineering: An industrial case study in the domain of oil and gas exploration and production. Int. J. Inf. Syst. Model. Des. 1(2), 1–22 (2010)
15. Guizzardi, G. et al.: Towards ontological foundations for conceptual modeling: The Unified Foundational Ontology (UFO) story. Appl Ontol 10(3-4), 259–271 (2015)
16. Lozano, J., Carbonera, J.L., Abel, M.: A novel approach for extracting well-founded ontology views. In: JOWO@ IJCAI (2015)
17. Lozano, J. at al.: Ontology view extraction: an approach based on ontological meta-properties. In: Proc. 26th IEEE ICTAI. pp. 122–129 (2014)
18. Moody, D.: The physics of notations: toward a scientific basis for constructing visual notations in software engineering. IEEE Trans. Softw. Eng. 35(6), 756–779 (2009)
19. Moody, D.L., Flitman, A.: A methodology for clustering entity relationship models: a human information processing approach. In: Proc.18th ER. pp. 114–130. Springer (1999)
20. Moody, D.L., Flitman, A.R.: A decomposition method for entity relationship models: A systems theoretic approach. In: Proc.ICSTM2000. vol. 72 (2000)
21. Sales, T.P., Guizzardi, G.: Ontological anti-patterns: Empirically uncovered error-prone structures in ontology-driven conceptual models. Data Know Eng 99, 72–104 (2015)
22. Snoeck, M.: Enterprise information systems engineering. The MERODE Approach (2014)
23. Teixeira, M.: An ontology-based process for domain-specific visual language design. Federal University of Espirito Santo, Brazil / Ghent University, Belgium (2016)
24. Tzitzikas, Y., Hainaut, J.L.: How to tame a very large er diagram (using link analysis and force-directed drawing algorithms). In: Proc.24th ER. pp. 144–159. Springer (2005)
25. Tzitzikas, Y., Kotzinos, D., Theoharis, Y.: On ranking rdf schema elements (and its application in visualization). J Univers. Comput Sci 13(12), 1854–1880 (2007)
26. Verdonck, M., Gailly, F.: Insights on the use and application of ontology and conceptual modeling languages in ontology-driven conceptual modeling. In: Proc.35th ER (2016)
27. Verdonck, M. et al.: Comparing traditional conceptual modeling with ontology-driven conceptual modeling: An empirical study. Inf Syst 81, 92–103 (2018)
28. Villegas Niño, A.: A filtering engine for large conceptual schemas. Universitat Politècnica de Catalunya (2013)