# Breaking into Pieces: An Ontological Approach to Conceptual Model Complexity Management

Guylerme Figueiredo
*NEMO, Computer Science Department*
*Federal University of Esprito Santo*
Vitoria, Brazil
gvsfigueiredo@inf.ufes.br

Amelie Duchardt
*Chair for Information Science*
*University of Regensburg*
Regensburg, Germany
amelie.duchardt@stud.uni-regensburg.de

Maria M. Hedblom
*CORE, Faculty of Computer Science*
*Free University of Bozen-Bolzano*
Bozen-Bolzano, Italy
mhedblom@unibz.it

Giancarlo Guizzardi
*CORE, Faculty of Computer Science*
*Free University of Bozen-Bolzano*
Bozen-Bolzano, Italy
gguizzardi@unibz.it

***Abstract*—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. 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 using the models' content. In other words, it is important that conceptual models are *cognitively tractable*. However, it is unavoidable that when the information of the represented domain grows, so does the size and complexity of the artifacts and models that represent them. For this reason, more sophisticated techniques for complexity management in ontology-driven conceptual models, need to be developed. Some approaches are based on the notion of *model modularization*. In this paper, we follow the work on model modularization to present an approach for *view extraction* for the ontology-driven conceptual modeling language OntoUML. We provide a formal definition for *ontological views* over OntoUML conceptual models that completely leverages on the ontologically well-grounded real-world semantics of that language. Moreover, we present a plug-in tool, particularly developed for an OntoUML model-based editor that implements this formal view structure in terms of queries defined over the OntoUML metamodel embedded in that tool.**

***Index Terms*—Conceptual Model Modularization, Ontological Views, Complexity Management in Conceptual Modeling, OntoUML**

## I. 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 held stimuli [20]. Therefore, there is an evident need for developing adequate engineering complexity management for reference conceptual models.

One type of such complexity management mechanisms is *Conceptual Model Modularization* (henceforth CMM), including the so-called *Ontology Modularization*. CMM is the process in which the model is fragmented into smaller interconnected parts [17]. The biggest challenge in CMM is the process for *module extraction*, namely, coming up with adequate criteria for dividing the model into modules.

Traditionally, different CMM techniques have been used for module extraction. However, almost the totality of these approaches address modularization in *ontologically-neutral languages* [12] such as UML, ER diagrams or OWL. These languages are either purely abstract syntax or have at most a formal (logical) semantics. As a result of the lack of *real-world semantics* in these modeling languages, the modularization techniques developed for them rely on criteria that leverage almost exclusively on the syntactical properties of the models, typically, topological ones [26].

In this paper, we take a different approach by developing a modularization technique that leverages on the ontological semantics behind the modeling constructs of the OntoUML language [10], [14]. OntoUML is an ontology-driven conceptual modeling language, whose modeling constructs and metamodel constraints reflect the ontological distinctions and axiomatization put forth by the Unified Foundational Ontology [10], [14]. OntoUML is employed here because it is the only conceptual modeling language that has explicitly defined real-world (ontological) semantics. Additionally, addressing this language is also of significant relevance given its growing base of users, who apply the language in a multitude of critical domains and in models of large complexity [23]. Moreover, as discussed in [11], OntoUML has been recently considered as a candidate for addressing OMG's Request for Proposal for *Semantic Model for Information Federation (SMIF)* and, as discussed therein, in scenarios of information federation,

we frequently have the conjunction of critical domains and models of significant complexity.

The contributions of this paper are two-fold: (i) firstly, we propose a structure of modules that can "break down" OntoUML models into model "pieces" that, on one hand, respect the underlying ontological semantics of the model elements involved and, on the other hand, strive to remain cognitively tractable. In this structure, the modules are formally defined as *Ontological Views*[1]; (ii) secondly, we provide an implementation of this structure in terms of queries defined over a metamodel embedded in a model-based OntoUML tool [9].

The remainder of the paper is organized as follows. Section II positions our work in reference to related efforts; Section III briefly presents the OntoUML language and some of the ontological notions underlying it; Section IV presents the first contribution of this paper by defining the aforementioned structure of ontological views. This is done both formally, in terms of a precise definition of the views, as well as intuitively by making use of a running example in the domain of Ship Transportation; Section V presents the second contribution of this paper, which is plug-in to a model-based editor in which this structure was implemented over the OntoUML metamodel. So this implementation itself counts as a first validation of the approach (claim to *practical realizability*). The section VI includes positioning our work in relation to previous approaches that closely resembles our method, such as the work of Lozano et al. [1], [17]; finally, Section VII presents some conclusions of the presented approach and some intended directions for future work.

## II. Related Work

The ideas behind conceptual model (including domain ontology) modularization have been around for some time and have been represented by a series of different approaches and techniques. This section aims to present a review of the state of the art in this area, both to contextualize the work presented here as well as putting it in contrast with other alternative approaches.

The problem of ontology modularization has received much attention in recent years. A number of approaches that aim to solve this problem have been proposed. One such approach was introduced by Khan [16]. His work introduces a framework that brings together a set of known modularization techniques. However, this work proposes more a methodology for modularization, without presenting any practical and automated approach to supporting that.

Villegas [26] provides a general overview of the techniques for model clustering. His approach proposes model-filtering mechanisms implemented as OCL queries over UML models. One limitation is that the approach solely relies on graph properties (e.g., visual distance, hierarchical distance, etc.) of the syntactical structures of the model.

In Dorans work [5] the extraction approach is based on domain ontologies following the work by Galen [21]. From a domain ontology codification, SPARQL queries are executed for module extraction. One limitation in this implementation is that the queries are connected to pre-configured domain ontologies, meaning that the approach is not generally applicable.

Seidenberg [24] proposes an approach for extracting parts of an ontology. However, the approach basically consists of syntactic rules, which means that the semantic relations between the involved elements are not fully considered therein. The rules are based on syntactical relations (e.g., generalization, part-of) or in the formal semantics of certain modeling constructs (e.g., transitivity of relationships).

Further work by Doran [6] focuses on extracting parts of an ontology for reuse by using an algorithm for traversing a graph. Here again, the technique completely relies on analyzing syntactical properties of a model as a graph.

Egyed [7] addresses an approach to automatic class model abstraction. Using the combination of some existing techniques, an algorithm for automatic abstraction was created. The algorithm is based on a set of rules that transforms the model into a graph and tries to infer abstractions of the model by means of the connectivity of the nodes. However, this approach has some limitations. One is the problem of treating multiple inheritance, since a relation of a subclass cannot always be abstracted to a relation with the superclass. As shown in [4], not supporting multiple classification is itself something that increases the complexity of models. Another problem is that, once more, the method relies on topological properties of the graph for analyzing the relevance of concepts in a model (e.g., the number of edges arriving in a node as a measure of domain relevance).

Coskun [3] proposes an approach for ontology partitioning through graph-based algorithms. The user instructs the ontology and the purpose to be analyzed by the tool. The ontology analyzer verifies the informed ontology while the configurator (a part of the system responsible for setting the goal, i.e. the user informs the goal and the configurator is responsible for the adjustments necessary to perform the goal) configures the tool according to the objective. Both these components inform the graph creator about the purpose and the analyzed structured ontology to create the graph. From the created graph, the partitioner uses the graph-based algorithms to partition the ontology.

We previously discussed, the aforementioned approaches were developed focusing on ontologically-neutral languages [12]. For this reason, they must rely either on purely syntactical properties of the models (e.g., topological properties of the

[1]Frequentely, in the literature, the term *modularization* is reserved to the process of breaking down a model into a set of non-overlapping parts but which together preserve the information content of the original artifact. In contrast, the term *view extraction* is typically used for the process of producing proper parts of the original model that are targeted at supporting specific tasks but that (being proper parts) do not preserve the information content of the original model. We here use the term modularization in a more liberal sense as a process of fragmenting the model into parts, each of which is an extracted view but which together preserve the information content of the original artifact. Perhaps a more appropriate name for what we are proposing here is *Conceptual Model Recoding*, for reasons that we will explain in section VII.

graph) or on basic formal semantic notions (e.g., formal meta-properties of relations such as transitivity or (a)symmetry). An approach that is an exception this rule is the one proposed by Lozano et al. in [1], [17]. In that approach, the authors develop a sub-ontology extraction algorithm, guided by ontological meta-properties for a new Visual Query System. This is the related work that is closest to our approach, not only because the authors make systematic use of ontological notions (e.g., identity, existential dependence, essential parthood) to develop their approach but, more specifically, because their filtering approach has also been developed for UFO and OntoUML.

## III. 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], [14]. UFO is an axiomatic formal theory based on contributions from Formal Ontology in Philosophy, Philosophical Logics, Cognitive Psychology, and Linguistics. OntoUML has been successfully employed in several industrial projects in different domains, such as petroleum and gas, digital journalism, complex digital media management, off-shore software engineering, telecommunications, retail product recommendation, and government [14]. A recent study shows that UFO is the second-most used foundational ontology in conceptual modeling and the one with the fastest adoption rate [25]. Moreover, the study also shows that OntoUML is among the most used languages in ontology-driven conceptual modeling (together with UML, (E)ER, OWL, and BPMN). 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. For an in-depth discussion, philosophical justifications, formal characterization and empirical support for these categories one should refer to [8], [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, Ship, and Harbor (see Figure 2). 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', likewise a kind 'Ship' can be specialized in the subkinds 'Cargo Ship' and 'Passenger Ship' (Figure 2).

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 properties of a person, in the way that 'being a puppy' captures a cluster of contingent properties of a dog, or in the way that 'being an active harbor' captures contingent properties of a harbor, see Figure 2) and *Roles* (for example, in the way that 'being a husband' captures a cluster of contingent properties of a man, or that 'being a captain' captures contingent properties of a person, see Figure 2). 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 active harbor' is being a harbor that is functional); 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; 'being a captain' is to bear a number of legal obligations and powers in the scope of a captain designation relationship to a ship, see Figure 2).

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 subclassification applied to the entities that "have that same type of essence".

Objects can relate to each other via parthood relations forming partonomic structure (e.g., a passenger ship can be composed of a business compartment and an economy compartment).

*Relators* (or relationships in a particular technical sense [8]) represent clusters of relational properties that "hang together" by a nexus (provided by a relator kind). Moreover, relators (e.g., marriages, enrollments, employments, presidential mandates, citizenships, but also transportation contracts, trips, administration assignments and captain designations, see Figure 2) 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 [8], 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].

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 harbor plays the role of a destination harbor in the scope of a trip, see Figure 2. 'Spouse' and 'President' (but also typically student, teacher, pet, destination harbor, captain, and traveling ship) 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). Another example is the role 'Ship Administrator' (which, again can be played by both people and organizations, see Figure 2). We call these role-like types that classify entities of multiple kinds *RoleMixins*.

In general, types that represent properties shared by entities of multiple kinds are termed *Non-Sortals*. In UFO, besides rolemixins, we have two other types of non-sortals, namely *Categories* and *Mixins*. Categories represent necessary properties that are shared by entities of multiple kinds (e.g., the category 'Physical Object' represent properties of all kinds of entities that have masses, spatial extensions, etc.). In contrast, mixins represent shared properties that are necessary to some of its instances but accidental to others (e.g., the mixin 'Red Object' can be thought as representing properties that are necessary to entities of certain kinds for instance, 'rubies', while being accidental to entities of other kinds for instance, 'apples'). Categories and mixins are, in contrast to rolemixins, considered as Relationally Independent Non-Sortals.

As previously mentioned, relators are existentially dependent entities. In fact, they are entities that are existentially dependent on a multitude of individuals. There entities that are, in contrast, existentially dependent on a single individual. These are termed here modes. Take, for instance, 'Johns Dengue Fever', 'Marys knowledge of Greek' or 'Pauls belief that Rome is the capital of Italy'. These are also full-fledged endurants that can exist in time maintaining their identity while changing in a qualitative manner (e.g., John's Dengue Fever can become a hemorrhagic fever) and which existentially depends on single individuals (e.g., John's Dengue Fever cannot exist without John existing). To connect modes to their bearers, OntoUML uses a relation of characterization (an exclusive type of existential dependence relation) [10].

## IV. Ontological Views

In this section, we present a formal definition of our structure of views. Built over UFO's distinctions and for the OntoUML language, the approach presented here propose rules to extract modules from a conceptual model or domain ontology expressed in OntoUML. As discussed in depth in [22] and formally demonstrated in [27], OntoUML is a pattern-language, i.e., the actual modeling primitives of this language are not primitives of low-granularity such as class, attribute, associations but clusters of constructs forming *ontology patterns*. As explained in the sequel, the definition of our ontological views follow closely the definition of these ontology patterns comprising the language. We advocate that these ontology patterns/views provide for a natural mechanism for "breaking" OntoUML models into cognitively tractable "pieces".

The Subsections A, B, and C below include some important pre-definitions to extract views, and the following items will describe formal rules for each ontology view. In the formal definition that follows, we here also approach an OntoUML model as a graph in which classes are nodes and relations (e.g., mediation, characterization, parthood and subtyping) are edges. The differences between this approach and the bulk of the literature in CMM are: (i) we consider that these graph elements have an interpretation (real-world semantics) defined in terms of a foundational ontology (UFO); (ii) we make systematic use of this ontological interpretation in order to defined the structure of our views.

### A. Basic Definitions

Let a Model M be a graph defined such that M=$< \Theta, \Sigma >$ and $\Theta$=$\{\mathbf{C_1..C_n}\}$ (a non-empty set of concepts in the model M) and $\Sigma$= $\{\mathbf{r_1..r_n}\}$ (set of directed relations in the model M). Let CT (Concept Type) and RT (Relation Type) be domains of types such that: CT = {**SORTAL, NON-SORTAL, KIND, SUBKIND, PHASE, ROLE, CATEGORY, ROLEMIXIN, MIXIN**}, RT = {**SUBTYPING, PARTHOOD, MEDIATION**}. Now, let $<$ be partial order relation defined in CT in the following way: **KIND** $<$ **SORTAL, SUBKIND** $<$ **SORTAL, ROLE** $<$ **SORTAL, PHASE** $<$ **SORTAL, CATEGORY** $<$ **NON-SORTAL, ROLEMIXIN** $<$ **NON-SORTAL, MIXIN** $<$ **NON-SORTAL**. Finally, we define a number of auxiliary functions as follows:

1) **C(M)** is a function that maps a model M to its associated set $\Theta$;
2) **R(M)** is a function that maps a model M to its associated set $\Sigma$;
3) **E HasType T** is a relation connecting an element E to a type T in the following manner: if E is a concept then $T \in CT$. Otherwise, if E is a relation then $T \in RT$. We should also add that for any two types T and T' such that T $<$ T', if (E HasType T) then (E HasType T');
4) **t(r)** is a function that maps a relation r to the target (destination) of that directed relation;
5) **s(r)** is the complementary function that maps a relation r to the source (origin) of that directed relation.

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 $t(r) \in C(M)$.

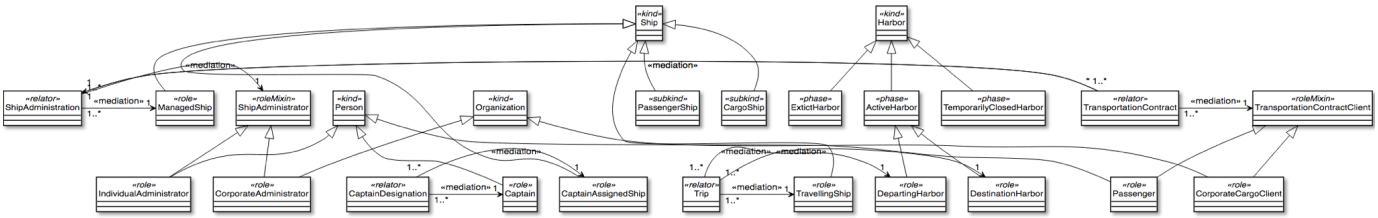


Figure 1. A Conceptual Model in the domain of Ship Transportation

For example, for the model of Figure 1, **C(M)** amounts to exactly the classes represented there, while **R(M)** includes all the mediation and UML subtyping relations.

### B. 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 **SUBTYPYING**) and such that (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'))
- 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: 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(Corporate Administrator) = Organization and K(Destination Harbor) = Harbor.

We can now define our views as follows:

### C. 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:

- $C(M) \subseteq C(M')$
- $R(M) \subseteq R(M')$

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 $t(r) \in C(M)$. In other words, M is necessarily an original subgraph of M'.

### D. Sortal Taxonomy View

We define that M is a taxonomic view of M' based on a salient base type T (symbolized as **STV(M,M',T)**, where (T $<$ **SORTAL**) iff:

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

STV is a generic parameterizable procedure that takes any sortal type and produces a view that includes classes of that type and the taxonomic structures of its supertypes (if any) until the level of kinds is reached.

### E. Kind View

We define that M is a Kind view of M' (symbolized as **KV(M,M')**) iff:

- STV(M,M',**KIND**)

One should notice that, in this case, STV will generate a set C(M) with exactly the kinds existing in model M' and $R(M) = \emptyset$.

The result of filtering the model of Figure 1 according to this definition, produces a view containing only the fundamental kinds of things that exist in that domain, namely, people, ships, organizations and harbors (see Figure 2). This view is constituted by terminal symbols of the OntoUML pattern language as defined in [22], [27].

### F. Subkind View

We define that M is a subkind view of M' (symbolized as **SKV(M,M')**) iff:

- STV(M,M',**SUBKIND**)

The result of filtering the model of Figure 1 according to this definition, produces a view that takes the two subkinds present in that model (PassengerShip and CargoShip) and produces a view that includes these subkinds and the classes and relations in this path until their (accidentally, in this case, common) kind is reached (see Figure 2).

This view is constituted by instances of the Subkind Pattern of the OntoUML pattern language as defined in [22], [27].

### G. Phase View

We define that M is a phase view of M' (symbolized as **PHV(M,M')**) iff:

- STV(M,M',**PHASE**)

The result of filtering the model of Figure 1 according to this definition, produces a view that takes the three phases present in that model (ExtinctHarbor, TemporarilyClosedHarbord and Active Harbor) and produces a view that includes these phases and the classes and relations in this path until their (accidentally, in this case, common) kind is reached (see Figure 2).

This view is constituted by instances of the Phase Pattern of the OntoUML pattern language as defined in [22], [27].

### H. Role View

We define that M is a subkind view of M' (symbolized as **RV(M,M')**) iff:

- STV(M,M',**ROLE**)

The result of filtering the model of Figure 1 according to this definition, produces a view that takes the the roles present in that model (see Figure 1) and produces a view that includes these roles and the classes and relations in this path until their respective kind is reached (see Figure 2).

This view is constituted by fragments that represent the core the Role Pattern of the OntoUML pattern language as defined in [22], [27]. These instances do not fully coincide with the instances of the entire role pattern because the latter includes a relational dependence pattern, which in our approach is dislocated to the Relational Context View.

### *I. Relational Context View*

We define that M is a phase view of M' based on a salient base type T (symbolized as **RCV(M,M',T)**) iff:

- V(M,M')
- $r \in R(M)$ iff ((r HasType T) OR ((r HasType **SUBTYPYING**) AND $s(r) \in C(M)$ AND $t(r) \in C(M)$))
- $c \in C(M)$ iff (there is a $r \in R(M)$ such that (r HasType T) AND (c = s(r) OR c = t(r)))

RCV is a generic parameterizable procedure that takes a relation type T and produces a view that includes classes related by relations of that type. It also includes subtyping relations connecting the types that are related by relations of type T.

### *J. Partonomy View*

We define that M is a partonomy view of M' (symbolized as **PV(M,M')**) iff:

- RCV(M,M',**PARTHOOD**)

PV is a relational context view in which the salient relation type T is a relation of parthood.

### *K. Mode View*

We define that M is a mode view of M' (symbolized as **MV(M,M')**) iff:

- RCV(M,M',**CHARACTERIZATION**)

MV is a relational context view in which the salient relation type T is a relation of characterization.

### *L. Relators and Mediation View*

We define that M is a phase view of M' (symbolized as **RMV(M,M')**) iff:

- RCV(M,M',**MEDIATION**)

RMV is a relational context view in which the salient relation type T is a relation of mediation. This view includes all relator types in the model as well as the mediation relations connecting them to other types in the model. Taking the model of Figure 1 as an example, we have the RMV depicted in Figure 2. In this view, we have the relators Transportation Contracts (connecting Transportation Contract Clients and Ship Administrations), Ship Administration (connecting Ship Administrators and Ships), Captain Designation (connecting Captain and Ship) and Trip (connecting Departing Harbor, Destination Harbor and Traveling Ship).

This view is constituted by instances of a version of the relator pattern of the OntoUML pattern language. This version includes the first three variants of this pattern as defined in [27] but also considers that relators can mediate endurants in general and not only objects (substantials).

### *M. Non-Sortal View*

We define that M is a non-sortal view of M' (symbolized as **NSV(M,M')**) iff:

- V(M,M')
- $c \in C(M)$ iff (c HasType **NON-SORTAL**) OR ((there is $c' \in C(M')$ such that (c' HasType **NON-SORTAL**)) AND (there is a c" such that (c" HasType **SORTAL**)) AND ST(c",c') AND IST*(c",c) AND IST*(c,K(c")))
- $r \in R(M)$ iff ((r HasType **SUBTYPING**) AND ($s(r) \in C(M)$) AND ($t(r) \in C(M)$))

The intention of the NSV can explained as follows. Take any non-sortal in the model (rolemixin, mixin or category), the view should include: (i) this non-sortal and all its non-sortal supertypes, including these subtyping relations connecting them; (ii) the first sortal specializing this non-sortal as well as the patch from this sortal to the unique kind providing its identity principle [10]. Taking the model of Figure 1 as an example, we have the NSV depicted in Figure 2. In this view, we have, for instance, the rolemixin ShipAdministrator, the sortals that immediately specialize it (the roles Individual Administrator and Corporate Administrator) 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).

The Non-Sortal view generated for our running example is constituted by two instances of the core fragment of the RoleMixin Pattern as defined in [22], [27]. These instances do not fully coincide with the instances of the entire RoleMixin pattern because the latter includes a relational dependence pattern, which in our approach is dislocated to the Relational Context View.

## V. A Tool for Ontological Views Extraction

In this section, we describe the tool for Ontological Views Extraction. This feature was built in *Menthor Editor*, an open-source ontology-driven conceptual modeling platform which incorporates the theories of Unified Foundational Ontology (UFO). The tool supports modeling, verification, validation, and implementation of OntoUML models.

Following the strategy proposed in Section IV, full support for pattern detection and view extractions have been implemented for this editor. In other words, by employing the explicitly defined MOF metamodel on which this editor is based, we have implemented algorithms to: automatically detect pattern occurrences, accessible through a detection dialog window (see example in Figure 3); extract views from OntoUML models comprising instances of these patterns. For instance, for the model of Figure 1, the tool will generate a structure of views that is equivalent to the one depicted in Figure 2.

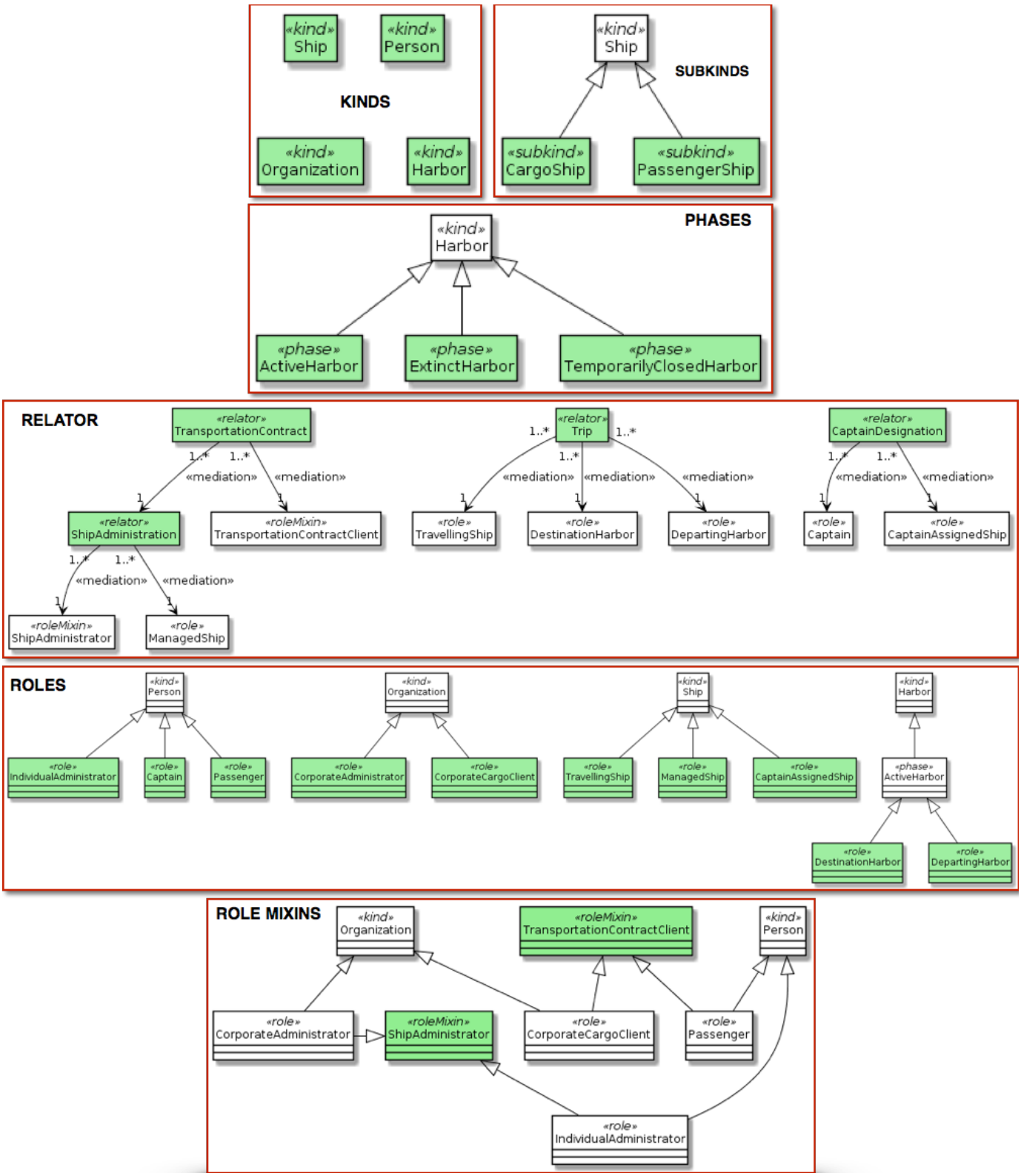


Figure 2. The modularized ship transportation model

OntoUML is a pattern-driven modeling language. As a consequence, we have that in an OntoUML model each element participates in at least one pattern. Moreover, the model elements can only appear in the models in the context of these patterns. Given that the ontological views as defined here are aggregation of these patterns from existing models, we have that the sum of generated views contains the same information as the original model. However, while preserving the information contained in the original model, our approach provides for a completely automated procedure that, by leveraging on the ontological semantics of the OntoUML language, breaks down that information in cognitively manageable chunks. These chunks are themselves composed of even finer-grained chunks, namely, the aforementioned ontology design patterns. As one can observe in Figure 2, these resulting building blocks stay within the threshold of human-cognitive capacity and manipulation in short-term memory [19].

## VI. Discussion

As discussed in Section II, most of the approaches presented there are restricted to using syntactic analysis techniques for conceptual model modularization. As a consequence, since the meanings of the represented elements are not taken into consideration, one risks semantically insensitive extraction results. In fact, most of the techniques presented in Section II are based on modeling languages that lack ontological expressiveness and, as such, are unable to differentiate between, for example, different sorts of types. For instance, when working with a UML class diagram, all types are simply represented as CLASSES and, as such, they stand in the same footing. In this way, it becomes significantly complex to identify domain relevance for the represented concepts.

To illustrate this point, take for example a class diagram in which we have an ADDRESS class and several other classes such as EMPLOYEE, ORGANIZATION and CLIENT that are connected to it (since conceptually all these types of entities can have one or more addresses). When this model is converted into a graph, a purely syntactical technique could consider that the most relevant node in that graph should be ADDRESS. However, concluding that this represents the most relevant concept in the domain would be mistake.

An undesirable conclusion such as this one is trivially avoided in our approach, since the different ontological categories applied to domain types is made explicit by the OntoUML stereotypes. So, here, different views are generated by focusing on different sorts of types that become salient for that view. So, for example, if we are interested in knowing the fundamental *Kinds* of things that exist in that domain, we can simply produce a view containing types with that sterotype (the *Kind View*); if we are interested in knowing in which sorts of *relationships* things of these kinds can participate and which roles they play in these relationships, we can simply produce a view organized around that information (a *Relator View* composed of instances of the Relator Pattern), and so on.

The approach presented by Lozano et al. [18] [17] uses the same modeling language (OntoUML) and the same underlying foundational ontology (UFO) considered in our work. For this reason, this is the approach in the literature that is of most relevance to the work presented here. The authors propose an algorithm that aims at producing models that: (1) preserve types carrying identity information and taxonomic relations; (2) find the identity Provider for types; (3) preserve qualities and existential dependency relations; (4) isolate essential parthood relations in the model; (5) preserve relational dependency as well as types related by formal relations. In this approach, the view extraction algorithm is based on a pre-selection of concepts informed by a user: firstly, the user selects one or more concepts in the model, then the algorithm identifies all elements in the model related to the selected ones such that together they maintain properties (1-5) above. The approach is effective in achieving its goals but strongly relies on users making sensible choices of pre-selected elements. In other words, if the elements selected by a user are not the most relevant for understanding the domain at hand, the view obtained from the extraction process would suffer from the same deficiency. In contrast, in the approach we are proposing here, the set of generated views is equivalent in terms of information to the complete model. Moreover, in our approach, the generated views are naturally constituted by the ontology design patterns of the OntoUML language. As a consequence, our approach automatically generates natural building blocks of finer-grained cognitively tractable chunks withing other cognitively tractable chunks.

## VII. Final Considerations

In this paper, we propose an approach for extracting views from conceptual models represented in the ontology-driven conceptual modeling language OntoUML. This approach makes systematic use of the real-world (ontological) semantics of this language to propose a structure of views that, on one hand, preserves all the information content of the original model but, on the other hand, ”breaks down” this information in different modules centered around different ontological concerns. These concerns can be loosely summarized as: (a) the fundamental *kinds* of things that exist in a domain; (ii) the *subkinds* into which these kinds of things are specialized; (iii) the *phases* that things of these kinds can go through; (iv) the fundamental types of relationships (*relators*) existing in that domain as well as the *roles* played in the scope of these relationships; (v) the kinds of things that can play each of these roles; (vi) the different abstract refactoring types that capture properties shared by things of multiple kinds (*non-sortals*); (vii) partonomical structures involving different types of entities in this domain; (viii) the modes that characterize different types of entities in this domain.

We have fully implemented this approach as a plug-in for a Model-Based OntoUML Editor. We have tested the scalability of this implementation against two artificially generated OntoUML models of large size. In the first test, against a model with 1317 classes, our implementation was able to compute the views and detect their constituting patterns in less than a second. In a second test, against a much larger model with

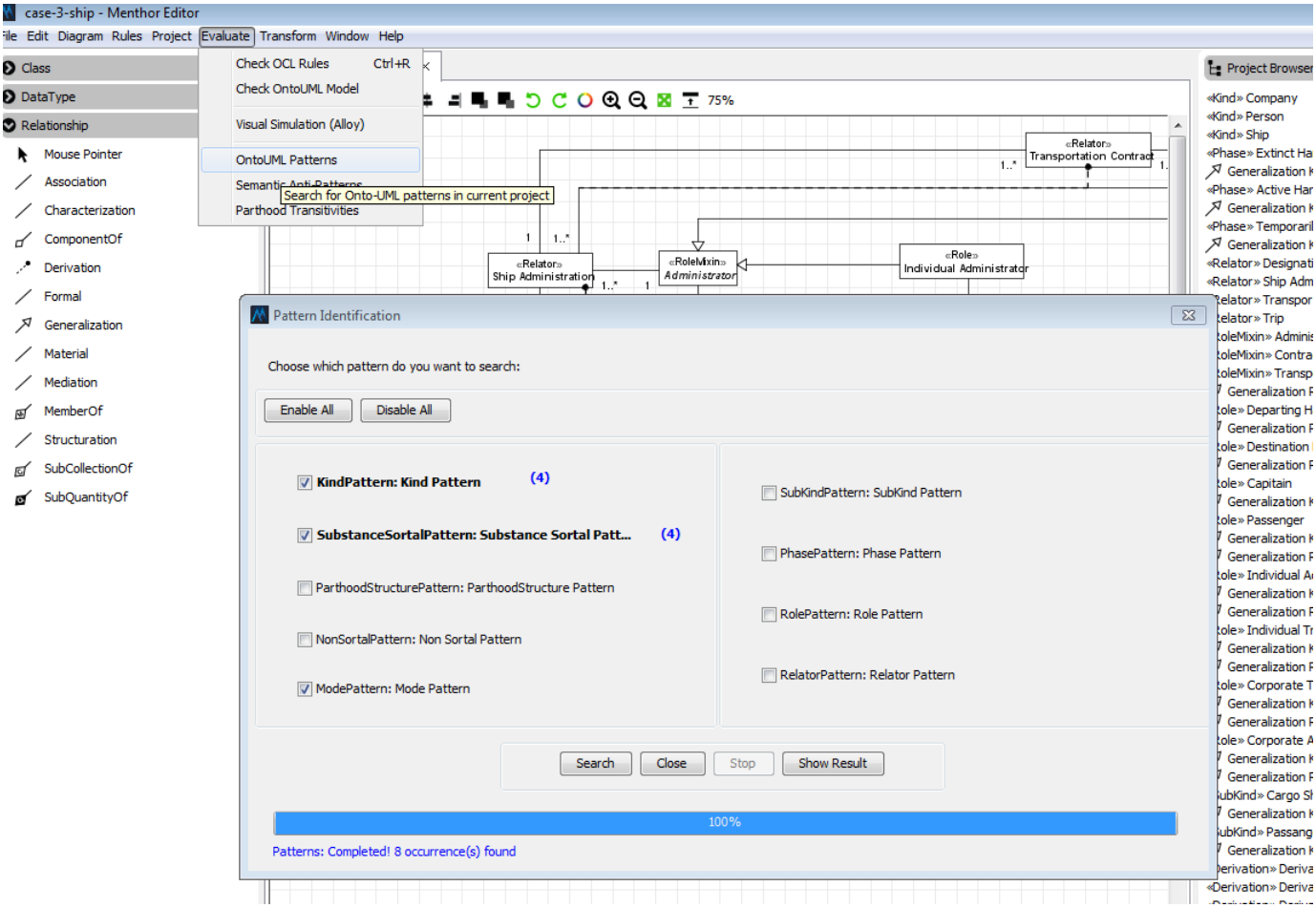


Figure 3. Screenshot of a Dialog Window for Pattern Detection in the Menthor Editor

12.474 classes, 6606 relations and 9504 subtyping relations, our implementation was able to perform the same operations in circa 1.8 minutes.

As one can observe in figure 2, the view extraction approach we propose here creates views that are composed of chunks derived from OntoUML ontology design patterns. In preliminary tests of our implementation against existing OntoUML models of different sizes and representing different domains, we observed that indeed the number of chunks within these views stay (in nearly the totally of cases) within the so-called *Miller's Magic Number* ($7 \pm 2$ items) [19]. As we previously mentioned, each of these chunks tend also to be cognitively tractable in this sense (i.e., tend not to significantly deviate from this limit number of subcomponents). Notice, however, that memory span is not limited in terms of the most basic presented ingredients but rather in terms of chunks, where a chunk is the largest meaningful unit in the presented material that the user recognizes. As put by Miller: "we must recognize the importance of grouping or organizing the input sequence into units or chunks. Since the memory span is a fixed number of chunks, we can increase the number of bits of information that it contains simply by building larger and larger chunks, each chunk containing more information than before" [19]. We advocate that grouping conceptual models in terms of the patterns organized within the views proposed here provide for ontology-based mechanism for "recoding" of conceptual models in Miller's sense. In fact, as we briefly mentioned before, a better term for what we are proposing here is perhaps *Conceptual Modeling Recoding* (as opposed to Modularization or View Extraction) as the striking feature of this process is to re-arrange conceptual models in higher-granularity chunks with high-semantic cohesion to improve model comprehensibility and recall but preserving information content.

As a follow up of this work, we plan to complete the definition of views proposed here to address all the ontology patterns and constructs of OntoUML [10], [22]. In fact, we intend to go even beyond the original version of the language to include new constructs and patterns that have been considered for the evolution of the language [8], [14] dealing, for example, with the modeling of events [13], [15] or multi-level structures [2]. We also plan to perform a more comprehensive and systematic series of tests for the plug-in implementation presented here.

Finally, we plan to empirically evaluate the adequacy of this approach. The very preliminary steps towards such a research program has been initiated with a pilot study that used

four models and their modularized equivalence, and included participants of different levels of conceptual modeling experience. While that pilot in itself was unable to provide concrete support for the hypothesis, it showed a trend that those not experienced with OntoUML had higher accuracy with the modularized ontology view. An additional consideration is to investigate the time duration needed to extract information from the models. Reasonably, it could be hypothesized that modularized models are easier to overview, and thus, also quicker to acquire information from. However, in order to confirm these hypotheses and the speculations of the efficiency of ontology modularization further investigations are required.

## ACKNOWLEDGMENTS

The authors are greatful to Alessander Botti Benevides and Daniele Porello for fruitful comments in the topics of this article.

## REFERENCES

[1] Aparicio, J.M.L.: Ontology view: a new sub-ontology extraction method (2015)
[2] de Carvalho, V.A., Almeida, J.P.A., Guizzardi, G.: Using a well-founded multi-level theory to support the analysis and representation of the powertype pattern in conceptual modeling. In: Advanced Information Systems Engineering - 28th International Conference, CAiSE 2016, Ljubljana, Slovenia, June 13-17, 2016. Proceedings. pp. 309–324 (2016)
[3] Coskun, G.: Structure-Based Partitioning of Semantic Web Ontologies. Ph.D. thesis, Freie Universität Berlin (2014)
[4] Deryck, M., Dvorák, O., Bruyn, P.D., Verelst, J.: Investigating the evolvability of financial domain models. In: Advances in Enterprise Engineering XI - 7th Enterprise Engineering Working Conference, EEWC 2017, Antwerp, Belgium, May 8-12, 2017, Proceedings. pp. 111–125 (2017)
[5] Doran, P., Palmisano, I., Tamma, V.A.: Somet: Algorithm and tool for sparql based ontology module extraction. WoMO 348 (2008)
[6] Doran, P., Tamma, V., Iannone, L.: Ontology module extraction for ontology reuse: an ontology engineering perspective. In: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management. pp. 61–70. ACM (2007)
[7] Egyed, A.: Automated abstraction of class diagrams. ACM Transactions on Software Engineering and Methodology (TOSEM) 11(4), 449–491 (2002)
[8] Guarino, N., Guizzardi, G.: "we need to discuss the relationship": Revisiting relationships as modeling constructs. In: Advanced Information Systems Engineering - 27th International Conference, CAiSE 2015, Stockholm, Sweden, June 8-12, 2015, Proceedings. pp. 279–294 (2015)
[9] Guerson, J., Sales, T.P., Guizzardi, G., Almeida, J.P.A.: Ontouml lightweight editor: A model-based environment to build, evaluate and implement reference ontologies. In: 19th IEEE International Enterprise Distributed Object Computing Workshop, EDOC Workshops 2015, Adelaide, Australia, September 21-25, 2015. pp. 144–147 (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: Conceptual Modeling - 33rd International Conference, ER 2014, Atlanta, GA, USA, October 27-29, 2014. Proceedings. pp. 13–27 (2014)
[12] Guizzardi, G., Baião, F.A., Lopes, M., de Almeida Falbo, R.: The role of foundational ontologies for domain ontology engineering: An industrial case study in the domain of oil and gas exploration and production. IJISMD 1(2), 1–22 (2010)
[13] Guizzardi, G., Guarino, N., Almeida, J.P.A.: Ontological considerations about the representation of events and endurants in business models. In: Business Process Management - 14th International Conference, BPM 2016, Rio de Janeiro, Brazil, September 18-22, 2016. Proceedings. pp. 20–36 (2016)
[14] Guizzardi, G., Wagner, G., Almeida, J.P.A., Guizzardi, R.S.S.: Towards ontological foundations for conceptual modeling: The unified foundational ontology (UFO) story. Applied Ontology 10(3-4), 259–271 (2015)
[15] Guizzardi, G., Wagner, G., de Almeida Falbo, R., Guizzardi, R.S.S., Almeida, J.P.A.: Towards ontological foundations for the conceptual modeling of events. In: Conceptual Modeling - 32th International Conference, ER 2013, Hong-Kong, China, November 11-13, 2013. Proceedings. pp. 327–341 (2013)
[16] Khan, Z.C., Keet, C.M.: An empirically-based framework for ontology modularisation. Applied Ontology 10(3-4), 171–195 (2015)
[17] Lozano, J., Carbonera, J., Abel, M., Pimenta, M.: Ontology view extraction: an approach based on ontological meta-properties. In: Tools with Artificial Intelligence (ICTAI), 2014 IEEE 26th International Conference on. pp. 122–129. IEEE (2014)
[18] Lozano, J., Carbonera, J.L., Abel, M.: A novel approach for extracting well-founded ontology views. In: JOWO@ IJCAI (2015)
[19] Miller, G.A.: The magical number seven, plus or minus two: Some limits on our capacity for processing information. The Psychological Review 63(2), 81–97 (March 1956)
[20] Moody, D.: The p̈hysics of notations: toward a scientific basis for constructing visual notations in software engineering. IEEE Transactions on Software Engineering 35(6), 756–779 (2009)
[21] Rogers, J., Rector, A.: The galen ontology. Medical Informatics Europe (MIE 96) pp. 174–178 (1996)
[22] Ruy, F.B., Guizzardi, G., de Almeida Falbo, R., Reginato, C.C., Santos, V.A.: From reference ontologies to ontology patterns and back. Data Knowl. Eng. 109, 41–69 (2017)
[23] Sales, T.P., Guizzardi, G.: Ontological anti-patterns: empirically uncovered error-prone structures in ontology-driven conceptual models. Data Knowl. Eng. 99, 72–104 (2015), https://doi.org/10.1016/j.datak.2015.06.004
[24] Seidenberg, J.: Web ontology segmentation: Extraction, transformation, evaluation. Modular Ontologies pp. 211–243 (2009)
[25] Verdonck, M., Gailly, F.: Insights on the use and application of ontology and conceptual modeling languages in ontology-driven conceptual modeling. In: Conceptual Modeling - 35th International Conference, ER 2016, Gifu, Japan, November 14-17, 2016, Proceedings. pp. 83–97 (2016)
[26] Villegas Niño, A.: A filtering engine for large conceptual schemas (2013)
[27] Zambon, E., Guizzardi, G.: Formal definition of a general ontology pattern language using a graph grammar. In: Proceedings of the 2017 Federated Conference on Computer Science and Information Systems, FedCSIS 2017, Prague, Czech Republic. pp. 1–10 (2017)