Science of Computer Programming 96 (2014) 417–443

Contents lists available at ScienceDirect

Science of Computer Programming

www.elsevier.com/locate/scico

Using a trope-based foundational ontology for bridging different areas of concern in ontology-driven conceptual modeling

Giancarlo Guizzardi [a,b,*], Veruska Zamborlini [c,d]

[a] Ontology and Conceptual Modeling Research Group (NEMO), Computer Science Department, Federal University of Espírito Santo (UFES), Brazil
[b] Laboratory for Applied Ontology (LOA), Institute for Cognitive Science and Technology (ISTC), Trento, Italy
[c] CR SANTEC, Centre de Recherche Public Henri Tudor, Luxembourg
[d] Vrije University Amsterdam, The Netherlands

HIGHLIGHTS

- We present a foundational ontology for conceptual modeling.
- We present a formal characterization of this ontology.
- We discuss a conceptual modeling language founded on this ontology (OntoUML).
- We provide mapping directives bridging OntoUML and the semantic web language OWL.
- We discuss computational support for model building, transformation and manipulation.

ARTICLE INFO

Article history:
Received 27 March 2013
Received in revised form 31 December 2013
Accepted 26 February 2014
Available online 11 March 2014

Keywords:
Ontological foundations for conceptual modeling
Conceptual domain modeling
Foundational ontology
Temporal reification
Trope theory

ABSTRACT

In recent years, ontology-driven reference models have gained much attention in the literature due to their potential key role in activities such as complex information modeling and semantic interoperability. The engineering process of these conceptual models should account for different phases addressing different areas of concern. In an initial phase of conceptual domain modeling, the target modeling artifacts should be constructed with the goal of maximizing quality attributes such as expressivity and truthfulness to the represented domain in reality. In a subsequent development phase, the resulting domain models can be used to guide the design decisions in the construction of different implementation artifacts addressing different computational concerns. In this paper, we present a philosophically sound, cognitively-oriented and formally characterized foundational theory of objects and tropes (property-instances). Moreover, we use this theory to bring about engineering contributions to both the aforementioned phases of ontology-driven conceptual modeling. Firstly, we show how this theory has been used to (re)design a system of modeling primitives underlying the conceptual domain modeling language OntoUML. Furthermore, we provide precise directives on how to map conceptual domain models in this language to their implementation in less-expressive computationally-oriented codification languages. In particular, we address here a mapping strategy to OWL (Web Ontology Language) that partially preserves the modal-temporal semantics of OntoUML. Finally, we discuss computational support for the proposed

* Corresponding author.
E-mail addresses: gguizzardi@inf.ufes.br (G. Guizzardi), v.carrettazamborlini@vu.nl (V. Zamborlini).

http://dx.doi.org/10.1016/j.scico.2014.02.022

approach in terms of conceptual model construction, automatic transformation and temporal querying.

1. Introduction

In December 2011, the Object Management Group (OMG) released a new Request for Proposal (RFP) entitled SIMF (Semantic Information Modeling Federation) [1]. The SIMF initiative is aimed at developing a "*standard that addresses the federation of information across different representations, levels of abstraction, communities, organizations, viewpoints, and authorities. Federation, in this context, means using independently conceived information sets together for purposes beyond those for which the individual information sets were originally defined*". Moreover, the proposal should "*define, adopt and/or adapt languages to express the conceptual domain models, logical information models and model bridging relationships needed to achieve this federation*".

Information Federation is inherently a semantic interoperability problem and underlying this RFP there is the recognition that current modeling technologies fall short in suitably supporting this task of semantic interoperability. At first, at the conceptual domain modeling level, we need a language that is truthful to the subtleties of the subject domains being represented. Moreover, this language should be expressive enough to make explicit the ontological commitment of the different worldviews underlying different models to be federated.

In a seminal paper [2], John Mylopoulos defines conceptual modeling as "*the activity of representing aspects of the physical and social world for the purpose of communication, learning and problem solving among human users*" and states that "*the adequacy of a conceptual modeling notation rests in its ability to promote understanding and problem solving regarding these domains among these human users ... not machines*". In summary, conceptual modeling is about representing (in diagrammatic notations) conceptualizations of reality to be shared among human users. For this reason, as defended by a number of authors over the years [3,4], a conceptual modeling notation should have its primitives grounded in the categories of a *Foundational Ontology*. Moreover, following the aforementioned desiderata, this Foundational Ontology should be one that takes both Cognition and *Linguistic Competence* seriously into consideration [5,6].

The expressivity in a modeling language needed to make explicit the ontological commitments of a complex domain tends to make this language prohibitive from a computational point of view in tasks such as automated reasoning. Conversely, computationally tractable logical languages tend to lack the expressivity to handle this essential aspect of semantic interoperability. For this reason, as defended in [5,6], we need to account for a two-step separation of concerns in domain modeling: (i) firstly, we should develop conceptual models as rich as possible to efficiently support the tasks of meaning negotiation and semantic interoperability across "*communities, organizations, viewpoints, and authorities*"; (ii) once the proper relationship between different information models is established, we can generate (perhaps several different) implementations in different logical languages addressing different sets of non-functional design requirements.

In this paper, we illustrate a number of the aforementioned aspects. Firstly, we present a fragment of a Cognitive Foundational Ontology, which has been employed over the years to analyze, re-design and integrate a number of conceptual modeling languages and reference models (Section 2). Secondly, we illustrate how this Foundational Ontology has been used to redesign an ontologically and cognitively well-founded Conceptual Domain Modeling Language (CDML) (Section 3). Finally, we show that this theory's ontological categories (which define the ontological semantics of this CDML) can be directly employed for creating transformations between models in this language and computationally-oriented representations (Section 4). In particular, we address here the issue of devising transformation strategies for representing the important modal (temporal) aspects of this CDML in OWL (Web Ontology Language), given the limitations of the latter language in representing this sort of information.

This article is an extension of [7], a complementary paper to an invited tutorial presented at the 5th International Conference on Software Language Engineering (SLE 2012), held in Dresden, Germany. In the original paper, we discussed informally the ontological theory on which the CDML OntoUML is based. Moreover, we laid out there the strategy for partially preserving the modal semantics of OntoUML when mapping its models to OWL. In this paper, in Section 2, we formally characterize some of the ontological distinctions comprising the fragment of this ontological theory. This formalization includes some novel aspects when compared to previous formalizations such as [8]. Additionally, in this paper, we present in full details the mapping directives between the fragment of OntoUML discussed here and OWL (Section 5). Moreover, in the same Section 5, we briefly discuss the existing tool support for the approach presented here in terms of a model-based editor whose embedded metamodel implements the modeling primitives of OntoUML, as well as the formal constraints derived from the formalization of the ontological theory presented in Section 2. In particular, we discuss a transformation embedded in this editor, which implements the mapping directives aforementioned, thus, fully automating the process of generating OWL specifications from OntoUML conceptual models following a temporal reification approach. Finally, to complete that section, we show how query patterns can be generated for retrieving temporal information from the resulting OWL specifications.

Finally, Section 6 of the article presents some final considerations.

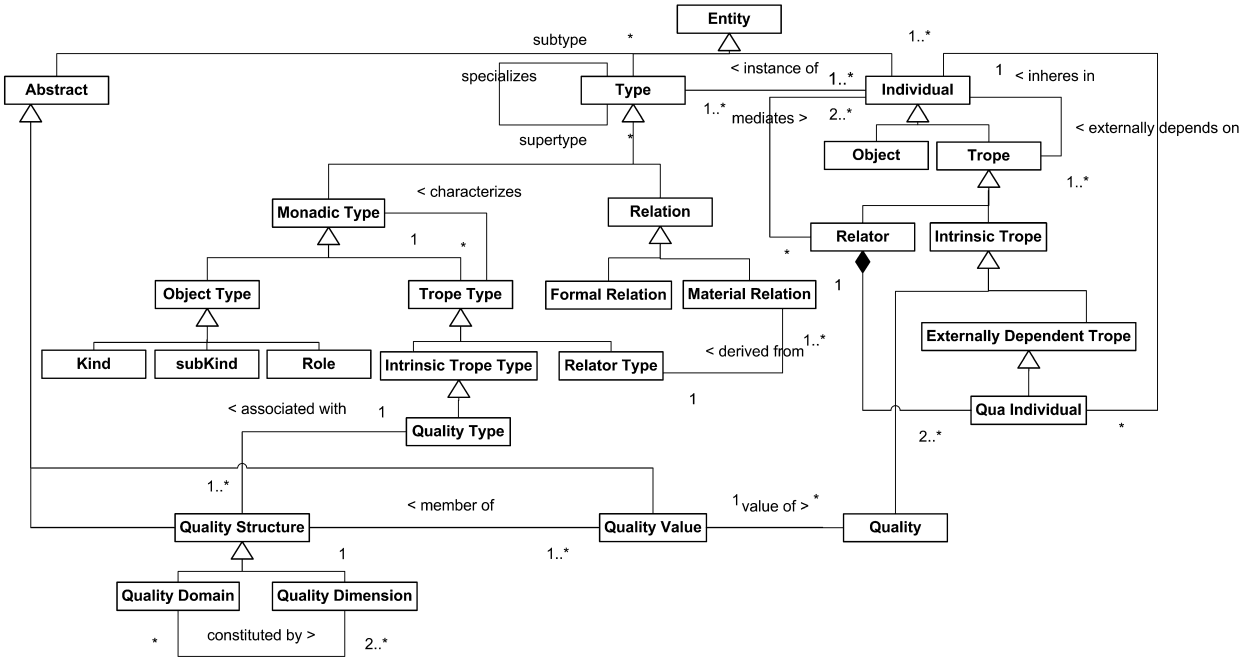

Fig. 1. A fragment of the Unified Foundational Ontology (UFO).

2. Ontological background

In this section, we discuss the Unified Foundational Ontology (UFO). UFO is a reference ontology based on a number of theories from Formal Ontology, Philosophical Logics, Philosophy of Language, Linguistics and Cognitive Psychology. In the sequel, we restrict ourselves to a fragment of this ontology, depicted in Fig. 1. Moreover, due to space limitations and the focus of the paper, we present the ontological categories comprising UFO superficially. For an in depth presentation and a full formalization of these categories, one should refer to [8].

2.1. Objects and tropes

A fundamental distinction in this ontology is between the categories of *Individual* and *Type* (or *Universals*). Individuals are entities that exist in reality possessing a unique identity. Types, conversely, are patterns of features that can be realized in a number of different individuals. The core of this ontology exemplifies the so-called *Aristotelian Ontological square* or what is termed a "*Four-Category Ontology*" [9] comprising the category pairs *Object–Object Type*, *Trope–Trope Type*. From a metaphysical point of view, this choice allows for the construction of a parsimonious ontology, based on the primitive and formally defined notion of *existential dependence*: We have that a particular x is *existentially dependent* (*ed*) on another particular y iff, as a matter of necessity, y must exist whenever x exists. Existential dependence is a modally constant relation, i.e., if x is dependent on y, this relation holds between these two specific particulars in all possible worlds in which x exists. Formally, we have that[1]:

$$ed(x, y) =_{def} \Box\big(\varepsilon(x) \rightarrow \varepsilon(y)\big)$$

Tropes (also termed *abstract particular*, *particular qualities*, *individual accident*, *mode, moment* or *property instance*) are existentially dependent entities.[2] Typical examples of tropes are: a color, a connection, an electric charge, a social commitment. There is solid evidence for tropes in the literature. On one hand, in the analysis of the content of perception [9,10], tropes such as colours, sounds, runs, laughter and singings are the immediate objects of everyday perception. On the other hand, the idea of tropes as truthmakers [10] underlies a standard event-based approach to natural language semantics, as initiated by Davidson [11] and Parsons [12].

[1] In the formalization employed in this article, we use the symbol $\Box$ to represent modal *necessity*. Thus, if A is a logical formula then $\Box A$ is true iff A is true in every possible world. In additional, as usual in modal logics formalizations, we use the symbol $\Diamond$ to represent modal *possibility*. Thus, if A is a logical formula then $\Diamond A$ is true iff there is a possible world in which A is true [13]. Moreover, we use the predicate ε here to represent existence in a world.

[2] In fact, in [8], we have employed the term *moment* as opposed to *trope*. We here use the term trope to maintain a terminological compatibility with [6,7] of which this paper can be considered an extension. Moreover, anecdotal evidence seem to show that the latter term is more neutral in its colloquial implications, thus, generating less confusion than the former.

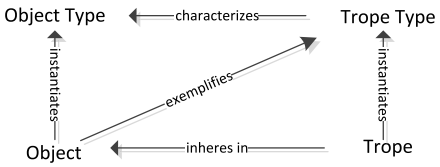

Fig. 2. The Aristotelian Square.

An important feature that characterizes all *tropes* is that they can only exist in other particulars termed their *bearers* (in the way in which, for example, my headache can only exist if I exist or that the marriage between John and Mary can exist if both of them exist). Existential dependence can also be used to differentiate intrinsic and relational tropes: *intrinsic tropes* are dependent of one single particular (e.g., color, a headache, a temperature); *relational tropes* (or *relators*) depend on a plurality of individuals (e.g., an employment, a medical treatment, a marriage). A special type of existential dependence relation that holds between a trope x and the particular y of which x depends is the relation of *inherence (i)*. Thus, for a particular x to be an intrinsic trope of another particular y, the relation $i(x, y)$ must hold between the two. For example, inherence glues your smile to your face, or the charge in a specific conductor to the conductor itself. Formally, inherence is an irreflexive, asymmetric and intransitive type of existential dependence relation:

$$\forall x, y \left(i(x, y) \rightarrow \mathit{Trope}(x) \wedge \mathit{Individual}(y)\right)$$
$$\forall x, y \left(i(x, y) \rightarrow \mathit{ed}(x, y)\right)$$
$$\forall x \neg i(x, x)$$
$$\forall x, y \left(i(x, y) \rightarrow \neg i(y, x)\right)$$
$$\forall x, y, z \left(i(x, y) \wedge i(y, z) \rightarrow \neg i(x, z)\right)$$

Inherence is also a functional relation, i.e., if h is the headache of John then it cannot be the headache of Paul. Thus, a trope must inhere in *exactly one individual*, which is termed its bearer (symbolized as β)[3]:

$$\mathit{Trope}(x) =_{def} \mathit{Individual}(x) \wedge \exists y\, i(x, y)$$
$$\forall x, y, z \left(\mathit{Trope}(x) \wedge i(x, y) \wedge i(x, z) \rightarrow y = z\right)$$
$$\beta(x) =_{def} \iota y i(x, y)$$

Here, we admit that tropes can inhere in other tropes. Examples include the individualized time extension, or the graveness of a particular symptom. The infinite regress in the inherence chain is prevented by the fact that there are individuals that cannot inhere in other individuals, namely, *Objects*.[4]

Examples of *objects* include ordinary entities of everyday experience such as an individual person, a dog, a house, a hammer, a car, Alan Turing and The Rolling Stones but also the so-called *Fiat Objects* such as the North-Sea and its proper-parts and a non-smoking area of a restaurant. In contrast with tropes, objects do not inhere in anything and, as a consequence, they enjoy a higher degree of independence. To state this precisely we say that: an object x is *independent* of all other objects that are disjoint from x, i.e., that do not share a common part with x. This definition excludes the dependence between an object and its *essential* and *inseparable parts* [8], and the obvious dependence between an object and its essential tropes.

To complete the Aristotelian Square, depicted in Fig. 2, we consider here the categories of *object type* and *trope type*. A trope type is also termed in the philosophical literature a *Property*. We also use the relation of *instantiation or classification* (symbolized as ::) between individuals and types. Object types classify objects and trope types classify tropes. Examples of the former include the types Apple, Planet and Person. Examples of the latter include the types Color, Electric Charge and Headache. We have that if an object x of type T bears a trope y of type Q_T then x is said to *exemplify* Q_T. Finally, if every instance x of T exemplifies Q_T then Q_T is said to *characterize* T. For instance, if we have that the type *Car* is characterized by the trope type *Chassis Number* then every particular car exemplifies the property of *having a chassis number* and does that by bearing a particular chassis number. Formally, we have that:

$$\mathit{ObjectType}(T) =_{def} \mathit{Type}(T) \wedge \forall x \left(x :: T \rightarrow \mathit{Object}(x)\right)$$
$$\mathit{TropeType}(Q_T) =_{def} \mathit{Type}(Q_T) \wedge \forall q \left(q :: Q_T \rightarrow \mathit{Trope}(q)\right)$$
$$\mathit{exemplifies}(x, Q_T) =_{def} \mathit{TropeType}(Q_T) \wedge \exists y \left(i(y, x) \wedge y :: Q_T\right)$$
$$\mathit{characterizes}(Q_T, T) =_{def} \mathit{Type}(T) \wedge \mathit{TropeType}(Q_T) \wedge \forall x \left(x :: T \rightarrow \mathit{exemplifies}(x, Q_T)\right)$$

[3] The iota operator (ι) used in a formula such as $\iota x \varphi$ was defined by Russel in [14] and implies both the existence and the uniqueness of an individual x satisfying predicate φ.

[4] As a synonym for the term *Object* as employed here, we have used the term *Substantial* elsewhere [8]. The variation, which is only terminological, is again motivated by the goal of harmonizing this theory with a jargon closer to the one already used by the conceptual modeling community.

Within the category of types, we make a fundamental distinction based on the formal notions of *rigidity* and *anti-rigidity*: A type T is rigid if for every instance x of T, x is necessarily (in the modal sense) an instance of T. In other words, if x instantiates T in a given world w, then x must instantiate T in every possible world w'. Formally, we have that:

$$Rigid(T) =_{def} \forall x \left(x :: T \rightarrow \Box\big(\varepsilon(x) \rightarrow x :: T\big)\right)$$

In contrast, a type T is anti-rigid if for every instance x of T, x is *possibly* (in the modal sense) not an instance of T. In other words, if x instantiates T in a given world w, then there must be a possible world w' in which x does not instantiate T:

$$AntiRigid(T) =_{def} \forall x \left(x :: T \rightarrow \Diamond\big(\varepsilon(x) \wedge \neg(x :: T)\big)\right)$$

2.2. Object types

In this article, we consider only a particular sort of object types termed *sortals* [22]. Sortals are types that are able to carry a uniform principle of identity, persistence and counting for all its instances. This is in contrast with *Mixins* (or non-sortals), which are abstract types that merely classify entities that share some (relational or intrinsic) properties but which are identified and individuated by different principles provided by different sortals [22]. For instance, take the mixin type *Insured Item*. This type does provides a principle of application for its instances, i.e., a principle with which we can judge if something is an instance of *Insured Item* (e.g., being included in an insurance policy). However, since the type classifies entities as persons, cars, civil constructions, works of art, body parts, among others, it cannot carry a uniform principle of identity to all its instances. Hence, it merely classify entities that instantiate different sortals (e.g., persons, cars, etc.).

A (sortal) object type that *supplies* the unique uniform principle of identity obeyed by all its instances is named a ***Kind***. A Kind is necessary rigid. After all, given that a principle of identity must apply to its instances in all possible situations it must be supplied by a type that is instantiated by these entities in every possible situation [22]. A rigid subtype of a Kind is named a ***subKind***.

An anti-rigid object type is termed here a ***Phased-Sortal*** [8]. The prototypical example highlighting the modal distinction between the categories Kind and Phased-Sortal is the difference between, on one hand, the Kind Person and, on the other hand, the Phase-Sortals Student and Adolescent instantiated by the individual John in a given circumstance. Whilst John can cease to be a Student and Adolescent (and there were circumstances in which John was not one), he cannot cease to be a Person. In other words, while the instantiation of the phase-sortals Student and Adolescent has no impact on the identity of a particular, if an individual ceases to instantiate the type Person, then he ceases to exist as the same individual.

In the example above, John can move in and out of the Student type, while being the same individual, i.e. without losing his identity. This is because the principle of identity that applies to instances of Student and, in particular, that can be applied to John, is the one that is supplied by the Kind Person of which the Phase-Sortal Student is a subtype. This is always the case with Phased-Sortals, i.e., for every Phased-Sortal PS, there is a unique ultimate Kind K, such that: (i) PS is a specialization of K; (ii) K supplies the *unique principle of identity* obeyed by the instances of PS. If PS is a Phased-Sortal and K is the Kind specialized by PS, there is a *specialization condition* φ such that x is an instance of PS iff x is an instance of K that satisfies condition φ. Formally, we have that[5]:

$$\forall T, T'\ specializes\big(T', T\big) \rightarrow Type(T) \wedge Type\big(T'\big) \wedge \Box\big(\forall x\, x :: T' \rightarrow x :: T\big)$$

$$\forall T\ ObjectType(T) \leftrightarrow Kind(T) \vee subKind(T) \vee PhasedSortal(T)$$

$$\forall T, T'\ ObjectType(T) \wedge specializes\big(T', T\big) \rightarrow ObjectType\big(T'\big)$$

$$\forall T\ ObjectType(T) \rightarrow Kind(T) \vee \exists! K\ Kind(K) \wedge specializes(T, K)$$

$$\forall K\ Kind(K) \rightarrow \neg\exists T\ specializes(K, T)$$

$$\forall K\ Kind(K) \vee subKind(K) \rightarrow Rigid(K)$$

$$\forall PS\ PhasedSortal(PS) \rightarrow AntiRigid(PS)$$

A particular type of Phased-Sortal emphasized in this article is what is named in the literature a ***Role***. A role Rl is an anti-rigid object type whose specialization condition φ is an extrinsic (relational) one. For example, one might say that if John is a Student then John is a Person who is enrolled in some educational institution, if Peter is a Customer then Peter is a Person who buys a Product x from a Supplier y, or if Mary is a Patient than she is a Person who is treated in a certain medical unit. In other words, an entity plays a role in a certain context, demarcated by its relation with other entities. This meta-property of Roles is named *Relational Dependence* and can be formally characterized as follows: A type T is relationally dependent on another type P via relation R iff for every instance x of T there is an instance y of P such that x and y are related via R [8].

[5] We use here for *uniqueness quantification* the well-known Kleene's quantifier ($\exists!$), which can be verbalized as "*there is exactly one*" and assume the specializes relation as a strict partial order relation [8].

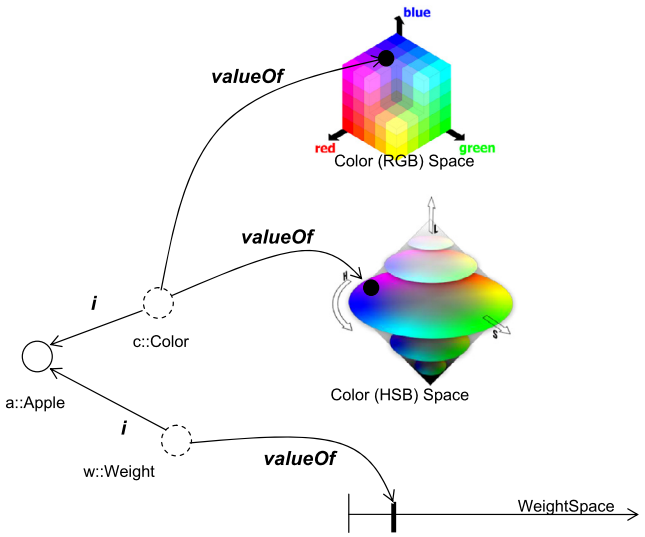

Fig. 3. An object, some of its inhering qualities and the associated quality structures.

2.3. Qualities and quality structure

An attempt to model the relation between intrinsic tropes and their representation in human cognitive structures is presented in the theory of *conceptual spaces* introduced in [15]. The theory is based on the notion of *quality dimension*. The idea is that for several perceivable or conceivable quality types there are associated quality dimensions in human cognition. For example, height and mass are associated with one-dimensional structures with a zero point isomorphic to the half-line of nonnegative numbers. Other properties such as color and taste are represented by multi-dimensional structures. Moreover, the author distinguishes between *integral* and *separable* quality dimensions: "*certain quality dimensions are integral in the sense that one cannot assign an object a value on one dimension without giving it a value on the other. For example, an object cannot be given a hue without giving it a brightness value* (...) *Dimensions that are not integral are said to be separable, as for example the size and hue dimensions.*" He then defines a *quality domain* as "a set of integral dimensions that are separable from all other dimensions" [15]. Furthermore, he defends that the notion of conceptual space should be understood literally, i.e., quality domains are endowed with certain geometrical structures (topological or ordering structures) that constrain the relations between its constituting dimensions. Finally, the perception or conception of an intrinsic aspect can be represented as a point in a quality domain. This point is named here a *quality value*.

Once more, an example of a quality domain is the set of integral dimensions related to color perception. A color quality c of an apple a takes its value in a three-dimensional color domain constituted of the dimensions hue, saturation and brightness. The geometric structure of this space (the *color spindle* [15]) constrains the relation between some of these dimensions. In particular, saturation and brightness are not totally independent, since the possible variation of saturation decreases as brightness approaches the extreme points of black and white, i.e., for almost black or almost white, there can be very little variation in saturation. A similar constraint could be postulated for the relation between saturation and hue. When saturation is very low, all hues become similarly approximate to grey.

We adopt in this work the term *quality structures* to refer to quality dimensions and quality domains, and we define the formal relation of *association* between a quality structure and an intrinsic trope type. Additionally, we use the terms *quality types* for those intrinsic trope types that are directly *associated with* a quality structure, and the term *quality* for a trope classified under a quality type. Furthermore, we define the relation of *valueOf* connecting a quality to its quality value in a given quality structure. Finally, we also have that quality structures are always associated with a unique quality type, i.e., a quality structure associated with the type Weight cannot be associated with the type Color. This is not to say, however, that different quality structures cannot be associated with the same quality type. For instance, with the quality type color, we can have both the HSB (Hue–Saturation–Brightness) structure and the RGB (Red–Green–Blue) structure. In Fig. 3, we illustrate an entity, its intrinsic color quality and the value of this quality mapped to into two different quality structures, hence, producing two different (albeit comparable) quality values.

In the sequel, we formalize some important aspects of the discussion of the previous paragraphs. Firstly, we have that quality types are those trope types whose instances are qualities. Moreover, we have the relation of $assoc(x, y)$ to represent a formal relation between and quality structure x and its associated quality type y, as well as the relation of $valueOf(v, q)$ to represent the formal relation between a quality q and its quality value v. Finally, we assume here that qualities instantiate a unique quality type and that these quality types are rigid:

$$QualityType(Q_T) =_{def} TropeType(Q_T) \wedge \forall x \left(x :: Q_T \rightarrow Quality(x)\right)$$
$$\forall Q_T \; QualityType(Q_T) \rightarrow Rigid(Q_T)$$

$$\forall x, y\ assoc(x, y) \rightarrow QualityStructure(x) \wedge QualityType(y)$$
$$\forall x, y\ valueOf(x, y) \rightarrow QualityValue(x) \wedge Quality(y)$$
$$\forall q\ Quality(q) \rightarrow \exists! Q_T\ QualityType(Q_T) \wedge (q :: Q_T)$$

Every quality value is a member of a unique quality structure. Moreover, every quality type is associated with a quality structure and quality structures are always associated with a unique quality type.

$$\forall x\ QualityValue(x) \rightarrow \exists! y\ QualityStructure(y) \wedge memberOf(x, y)$$
$$\forall x\ QualityStructure(x) \rightarrow \exists! y\ QualityType(y) \wedge assoc(x, y)$$
$$\forall x\ QualityType(x) \rightarrow \exists y\ QualityStructure(y) \wedge assoc(y, x)$$

We have that a quality has exactly one quality value in each of the quality structures associated with the quality type it instantiates:

$$\forall q, Q_T\ (q :: Q_T) \wedge QualityType(Q_T) \rightarrow$$
$$\big(\forall Q_S\ assoc(Q_S, Q_T) \rightarrow \exists! v\ memberOf(v, Q_S) \wedge valueOf(v, q)\big)$$

But also that if v is the value of a quality q then it is the value of q in one quality structure associated with the quality type instantiated by q

$$\forall v, q\ valueOf(v, q) \rightarrow \big(\exists Q_T, Q_S\ assoc(Q_S, Q_T) \wedge q :: Q_T \wedge memberOf(v, Q_S)\big)$$

Finally, we have that:

$$\forall q, v\ valueOf(v, q) \rightarrow \Box\big(valueOf(v, q)\big)$$

Now, suppose that we have the type T is characterized by the quality type Q_T, and that we have the quality structure Q_S associated with the quality type Q_T. We can then define the following function schema (named an *attribute function schema*, or simply, *attribute schema*):

$$[x \triangleright y] : A(T, Q_S) \leftrightarrow x :: T \wedge memberOf(y, Q_S) \wedge \exists q\ \big(i(q, x) \wedge valueOf(y, q)\big)$$

In this attribute schema, the quality type Q_T does not appear explicitly. However, given the formulae presented above, it must be the quality type instantiated by q and associated with Q_S. Take, for instance, the object type Apple characterized by the quality type weight. In this case, we have that:

$$\forall a\ \big(a :: Apple \rightarrow \exists w\ \big(w :: Weight \wedge i(w, a)\big)\big)$$

Thus, the mapping between an apple a and its weight quality value in a particular quality structure (associated with the Weight quality type) can be represented by the following *attribute* (a particular configuration of an attribute schema):

$$[a \triangleright wv] : att\text{-}weight(Apple, WeightValues) \leftrightarrow$$
$$a :: Apple \wedge memberOf(wv, WeightValues) \wedge \exists w\ \big(i(w, a) \wedge valueOf(wv, w)\big)$$

The view of qualities defended here assumes that change is *replacement* (as opposed to variation) of tropes, i.e. "the color of x turned from red to brown" is represented by a red-quality of x that temporally precedes a brown-quality of x. As a consequence, we have that although a quality q can have different quality values in different quality spaces, their values in each of these structures cannot be changed. Taking this view into consideration, we elaborate further in two orthogonal dimensions capturing specific characteristics of qualities, which are related to aspects of temporal change of their bearers.

In the first dimension, we distinguish between **necessary** (mandatory) versus **contingent** (optional) properties; in the second, we distinguish between **immutable** versus **mutable** ones. The former distinction refers to the need for an entity to bear that property, regardless of its value. For instance, suppose that in a given conceptualization of (legal) Person, both *name* and *age* are mandatory characteristics of people (every person has a name and an age) whilst *ssn* (*social security number*) and *nickname* (*alias*) are, in contrast, optional characteristics of people. Now, notice that the relation between a person and age is a relation of *generic dependence*, i.e., the same person can bear different age qualities in different situations as long as they are all instances of the quality type (property) *age*. This brings us to the second of these partitions: a quality q is immutable to a bearer x of type T iff x must bear that very same quality in all possible situations in which x instantiates T. In this case, the relation between x and q is a relation of *specific dependence* (as opposed to a generic one). Again, let us suppose that, in a given conceptualization, (legal) persons cannot change their proper names. In this situation, a name would not only be a necessary but also an immutable characteristic of people. Suppose now that, in this conceptualization, that although *ssn* is an optional characteristic of people, once an *ssn* is assigned to a person, it cannot be changed. In this case, *ssn* would be an immutable and contingent property. Finally, in this conceptualization, we assume that nicknames are both optional to people and, once assigned, can always be changed. In this case, nickname would be an

example of a contingent and mutable property. In the sequel, we formally elaborate on the relation of *characterization* in Fig. 2 by taking into account these two dimensions.

From a formal point of view, the case of an optional property can simply be characterized as follows. We state that a trope type Q_T *optionally characterizes* a type T by:

$$opt\text{-}charac(Q_T, T) =_{def} Type(T) \wedge TropeType(Q_T) \wedge \forall x\,(x :: T \rightarrow \Diamond(\exists y\, y :: Q_T \wedge i(y, x))$$

In contrast, we state that a type Q_T *mandatorily characterizes* a type T as:

$$mand\text{-}charac(Q_T, T) =_{def} Type(T) \wedge TropeType(Q_T) \wedge \forall x\,(x :: T \rightarrow \Box(\varepsilon(x) \wedge x :: T \rightarrow \exists y\, y :: Q_T \wedge i(y, x)))$$

As one can notice *mandatorily characterizes* implies *optionally characterizes*. In other words, if an instance x of T exemplifies a certain property Q_T in every possible situation in which x is an instance of T then there is a situation in which x exemplifies Q_T.

Regarding immutability, we can characterize an immutable property as follows. We state that trope type Q_T *immutably characterizes* a type T by:

$$im\text{-}charac(Q_T, T) =_{def} Type(T) \wedge TropeType(Q_T) \wedge \forall x\,(x :: T \rightarrow \exists y\, y :: Q_T \wedge \Box(\varepsilon(x) \wedge x :: T \rightarrow i(y, x))$$

Again, one can notice *immutably characterizes* implies *mandatorily characterizes*. In other words, if an instance x of T exemplifies a certain property Q_T in every possible situation that x instantiates T and always with the same value then x exemplifies Q_T in every possible situation that it instantiates T.

Finally, the case of mutable characterization can be expressed as follows:

$$mut\text{-}charac(Q_T, T) =_{def} opt\text{-}charac(Q_T, T) \wedge \forall x, y\,(x :: T \wedge y :: Q_T \wedge i(y, x)) \rightarrow \Diamond(\varepsilon(x) \wedge \neg i(y, x))$$

In other words, a type Q_T mutably characterizes a type T iff: it is possible for instances of T to exemplify Q_T; for every instance x of T that exemplifies Q_T by bearing an instance q of Q_T, we have that there is a counterfactual situation w in which x does not bear q (either because x stops exemplifying Q_T in w or because x bears another instance q' of Q_T in w).

2.4. Relators, relations, roles and qua individuals

Following the philosophical literature, we recognize here two broad categories of relations, namely, *material* and *formal* relations [16,17]. Formal relations hold between two or more entities directly, without any further intervening individual. Examples include the relations of *existential dependence* (*ed*), *subtype*, *instantiation*, *parthood, inherence (i)*, among many others not discussed here [8]. Domain relations such as *working at*, *being enrolled at*, and *being the husband of* are of a completely different nature. These relations, exemplifying the category of *material relations*, have material structure of their own. Whilst a formal relation such as the one between Paul and his headache x holds directly and as soon as Paul and x exist, for a material relation of *being treated in* between Paul and the medical unit MU_1 to exist, another entity must exist that *mediates* Paul and MU_1. These entities are termed *relators*.

Relators are individuals with the power of connecting entities. For example, a medical treatment connects a patient with a medical unit; an enrollment connects a student with an educational institution; a covalent bond connects two atoms. The notion of relator is supported by several works in the philosophical literature [8] and, they play an important role in answering questions of the sort: what does it mean to say that John is married to Mary? Why is it true to say that Bill works for Company X but not for Company Y? Again, relators are special types of tropes, which, therefore, are existentially dependent entities. The relation of *mediation* (symbolized m) between a relator r and the entities r connects is a sort of (non-exclusive) inherence and, hence, a special type of existential dependence relation.

An important notion for the characterization of relators (and, hence, for the characterization of material relations) is the notion of *foundation*. Foundation can be seen as a type of *historical dependence* [18,19], in the way that, for instance, an instance of *being kissed* is founded on an individual *kiss*, or an instance of *being punched by* is founded on an individual *punch*, an instance of *being connected to* between airports is founded on a particular flight connection. Suppose that John *is married to* Mary. In this case, we can assume that there is an individual relator m_1 of type *marriage* that mediates John and Mary. The foundation of this relator can be, for instance, a wedding event or the signing of a social contract between the involved parties. In other words, for instance, a certain event e_1 in which John and Mary participate can create an individual marriage m_1, which is existentially depends on John and Mary and, hence, mediates them. The event e_1 in this case is the foundation of relator m_1.

Now, let us elaborate on the nature of the relator m_1. There are many intrinsic tropes that John acquires by virtue of being married to Mary. For example, imagine all the legal responsibilities that John has in the context of this relation. These newly acquired properties are intrinsic tropes of John, which, therefore, are existentially dependent on him. However, these tropes also depend on the existence of Mary. We name this type of trope *externally dependent tropes*, i.e., an externally dependent trope q is an intrinsic trope that inheres in an individual x but that is also existentially dependent on at least another individual y, such that y is itself independent of x. Formally, we have that:

$$ExtDepTrope(q) =_{def} IntrinsicTrope(q) \wedge \exists y\ indep\big(y, \beta(q)\big) \wedge ed(q, y), \quad \text{where}$$
$$indep(x, y) =_{def} \neg ed(x, y) \wedge \neg ed(y, x)$$

The individual that is the aggregation of all externally dependent tropes that John acquires by virtue of being married to Mary is named a *qua individual* (in this case, *John-qua-husband-of-Mary*). A qua individual is, thus, defined as an individual composed of all externally dependent tropes that inhere in the same individual and share the same foundation. In the same manner, by virtue of being married to John, Mary bears an individual *Mary-qua-wife-of-John*.

The notion of qua individuals is the ontological counterpart of what has been named *role instance* in the literature [20] and represent the properties that characterize a particular mode of participation of an individual in a relation. Now, the entity that is the sum of all qua individuals that share the same foundation is a relator. In this example, the relator m_1 that is the aggregation of all properties that John and Mary acquire by virtue of being married to each other is an instance of the relational property *marriage*. The relation between the two qua individuals and the relator m_1 is an example of formal relation of parthood (*partOf*) [8].

The relator m_1 in this case is said to be the *truthmaker* of propositions such as "John is married to Mary", "Mary is married to John", "John is the husband of Mary", and "Mary is the wife of John". In other words, material relations such as *being married to*, *being legally bound to*, *being the husband of* can be said to hold for the individuals John and Mary because and only because there is an individual relator marriage m_1 mediating the two. Thus, as demonstrated in [8,19], material relations are purely linguistic/logical constructions, which are founded on and can be completely derived from the existence of relators. In fact, in [8], we have defined a formal relation of derivation (symbolized as *der*) between a relator type (e.g., Marriage) and each material relation that is derived from it.

Now, let x, y and z be three distinct individuals such that: (a) x is a relator; (b) y is a qua individual and y is part of x; (c) y inheres in z. In this case, we say that x *mediates* z. Formally we have that:

$$\forall x, y\ mediates(x, y) \rightarrow Relator(x) \wedge Individual(y)$$
$$\forall x\ Relator(x) \rightarrow \forall y\ \big(mediates(x, y) \leftrightarrow \big(\exists z\ quaIndividual(z) \wedge partOf(z, x) \wedge i(z, y)\big)\big)$$

Additionally, we require that a relator mediates at least two distinct individuals, i.e.,

$$\forall x\ Relator(x) \rightarrow \exists y, z\ \big(y \neq z \wedge mediates(x, y) \wedge mediates(x, z)\big)$$

Furthermore, there is an intimate connection between qua individuals and *role types*: let T be a natural type (kind) instantiated by an individual x, and let R be a role type specializing T. We have that there is a qua individual type Q_{ua} such that x instantiates R iff x bears an instance of Q_{ua}:

$$\forall R\ Role(R) \rightarrow \exists QuaT\ QuaIndividualType(QuaT) \wedge mand\text{-}charac(QuaT, R)$$

Notice that, in order to instantiate a role, an object must bear a qua individual. Moreover, given that qua individuals are parts of relators, we have here a conformance to the formal property of roles as *relationally dependent* types.

One issue that deserves a special attention at this point is the issue of immutable characterization when applied to anti-rigid types. For instance, suppose that the Role Student is defined as a subtype of Kind Person and it is defined to have an immutable property *student-id*. What does it mean then to state that student-id is immutable? According to the definition given for immutable characterization in Section 2.3, let *im-charac*(*Student-Id*, *Student*), then if we have that a particular student John bears a particular student-id (e.g., "st-id-123") then it must bear that student-id in every situation that John is a student. This seems to be too restrictive, after all, we can easily imagine John ceasing to be a student in a given institution and starting later in another institution with a different student-id. Trying to characterize John's connection to the st-id-123 as one of temporal constancy will also not suffice (e.g., while John's studentship with id *st-id-123* lasts, this id cannot be changed). Two obvious reasons are: John can be a student in several different institutions at the same time (with different yet immutable student ids in each of them); one studentship of John in a given institution can be interleaved with another studentship of John in a different institution. In other words, there is no guarantee that the time interval in which a certain studentship holds is convex. With the ontological framework proposed here, this problem can be addressed by recognizing that the immutable property student-id is in fact a property of the qua entity John-qua-student or a resultant property of relator $Enr_{John\text{-}U1}$ connecting John to a given university (U_1).

Finally, given a relator type defined as follows:

$$relatorType(R_T) =_{def} TropeType(R_T) \wedge \forall r\ \big(r :: R_T \rightarrow Relator(r)\big)$$

We have the relation of derivation (*der*) between a material relation φ and relator type R_T defined as such that an n-uple $\langle x_1 \ldots x_n \rangle$ instantiates φ iff there is a relator $r :: R_T$ and such that the following is true: $\bigwedge_{i \leqslant n} mediates(r, x_i)$.

The summary of the discussion promoted in this section is illustrated in Figs. 4–6. Fig. 4 illustrates the inherence relation between John and his externally dependent tropes, which are existentially dependent on Mary (as well as analogous relations in the converse direction). In Fig. 5, John instantiates the role type Husband (which is a specialization of the kind (Male) Person) iff there is a qua individual John-qua-husband-of-Mary which inheres in John. Moreover, this figure illustrates that the qua individuals John-qua-husband-of-Mary and Mary-qua-wife-of-John are mutually existentially dependent.

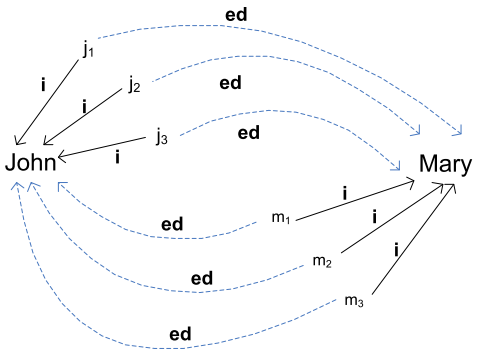

Fig. 4. Objects and their inhering externally dependent tropes: in this example, the object bears a number of tropes (j_1, j_2, j_3), which inhere (i) in John but, which are also existentially dependent (ed) on Mary. Mutatis Mutandis, the model depicts a number of tropes of Mary (m_1, m_2, m_3), which inhere in Mary but, which are also existentially dependent on John.

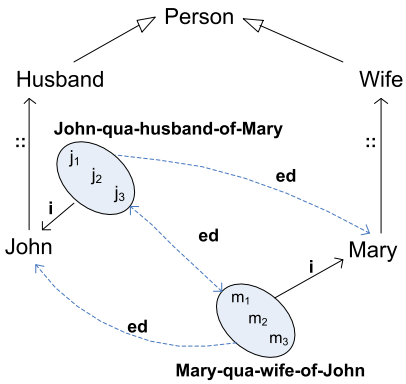

Fig. 5. Objects, their instantiating roles and their inhering qua individuals: in this example, John and Mary instantiate (::) the roles Husband and Wife, respectively, in virtue of the qua individuals that inhere (i) in them. These roles are specializations of the type Person ($\rightarrow$). Moreover, *John-qua-husband-of-Mary* (which is an aggregations of the tropes j_1, j_2 and j_3) is mutually existentially dependent (ed) on *Mary-qua-wife-of-John* (an aggregation of tropes m_1, m_2 and m_3).

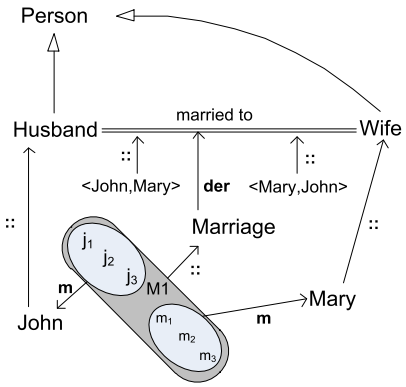

Fig. 6. Material relations are founded on relators that mediate their relata: in this example, the marriage relator M_1 between John and Mary mediates (m) these two entities by virtue of being existentially dependent on both of them. This relator is an aggregation of the qua individuals *John-qua-husband-of-Mary* and *Mary-qua-wife-of-John* (represented by the two ellipses). Moreover, M_1 is the foundation for the tuples ⟨John, Mary⟩ and ⟨Mary, John⟩, which instantiate (::) the material relation *married to*, which, in turn, is derived (der) from the relator type Marriage that M_1 instantiates.

In other words, John cannot be the Husband of Mary without Mary being the wife of John. Finally, Fig. 6 shows that the material relation *married to* is derived from the relator type Marriage and, thus, tuples such as ⟨John, Mary⟩ and ⟨John, Mary⟩ are instances of this relation iff there is an instance of Marriage that mediates the elements of the tuple.

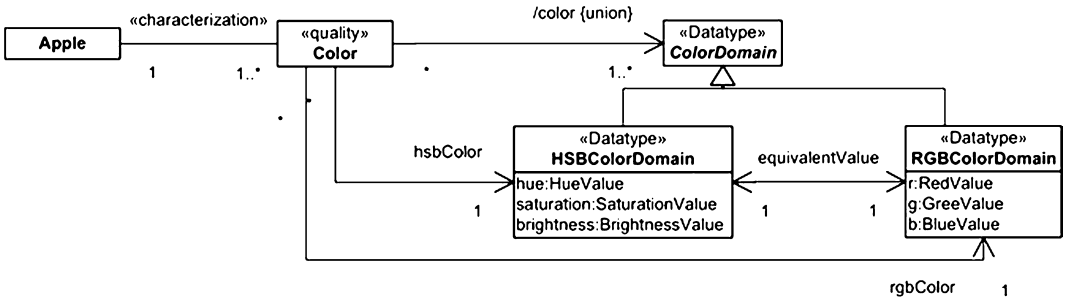

Fig. 7. Representing quality types that can be associated to multiple quality structures [21].

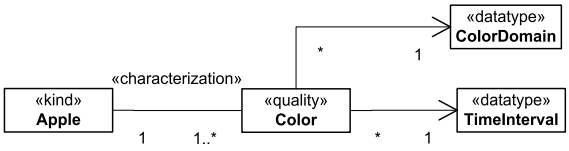

Fig. 8. Temporal change in properties as quality replacement.

3. Using a foundational ontology to design a well-founded conceptual modeling language

In this section, we briefly discuss a Conceptual Domain Modeling language termed OntoUML [8]. OntoUML is an example of a conceptual modeling language whose metamodel has been designed to comply with the ontological distinctions and axiomatization of the UFO foundational ontology.

The OntoUML metamodel contains: (i) elements that represent ontological distinctions prescribed by an underlying foundational ontology; (ii) constraints that govern the possible relations that can be established between these elements. These constraints, which are derived from the axiomatization of the ontological theory, restrict the ways in which the modeling primitives can be related. The goal is to have a metamodel such that all grammatically correct specifications according to this metamodel have logical models that represent *intended state of affairs* of the underlying conceptualization [5].

For instance, the language has modeling primitives to explicitly represent the notions of *kinds*, *subkind*, *roles* and *relator* previously discussed. Kinds and subkinds are represented by the corresponding stereotypes «kind» and «subkind».[6] In an analogous manner, roles are represented by the stereotype «role». In the axiomatization of the UFO ontology we have that anti-rigid types cannot be a supertype of rigid one [8]. So, as an example of formal constraint in this language, we have that classes stereotyped as «kind» or «subkind» cannot appear in an OntoUML model as a subtype of class stereotyped as «role». Additionally, following the formal characterization presented in Section 2, a class stereotyped as «role» must have as a supertype exactly one class stereotype as «kind».

As discussed at length in [21], quality types are typically not represented in a conceptual model explicitly but via *attribute functions* that map each of their instances to points (quality values) in a quality structure. Accordingly, the *datatype* associated with an attribute A of class C is the representation of the quality structure that is the co-domain of the attribute function represented by A. In other words, a quality structure is the ontological interpretation of the (Onto)UML datatype construct. Moreover, we have that a multidimensional quality structure (quality domain) is the ontological interpretation of the so-called *structured datatypes*. Quality domains are composed of multiple integral dimensions. This means that the value of one dimension cannot be represented without representing the values of others. The fields of a datatype representing a quality domain QD represent each of its integral quality dimensions. Alternatively, we can say that each field of a datatype should always be interpreted as representing one of the integral dimensions of the QD represented by the datatype. The constructor method of the datatype representing a quality domain must reinforce that its tuples always have values for all the integral dimensions. Finally, an algebra (as a set of formal constraints) can be defined for a datatype so that the relations constraining and informing the geometry of represented quality dimensions are also suitably characterized.

There are, nonetheless, two situations in which one might want to represent quality types explicitly. The first of these is when we want to represent that a quality might be projected to different quality spaces (i.e., the underlying quality type is associated with alternative quality structures). This idea is represented in Fig. 7. In this case, the color quality aggregates the different values (in different quality spaces) that can be associated with that object (the apple, in this case). In these situations, it is as if the color quality is representing a certain 'aspectual slice' of the Apple, or the *Apple-qua-colored object*.

A second situation in which one might want to represent qualities explicitly is when modeling a temporal perspective on qualities. This is illustrated in Fig. 8. In that model, we have different color qualities (with different associated quality values) inhering in a given apple in different situations.

[6] These stereotypes are precisely defined in the OntoUML metamodel ([8], p. 316) in terms of the homonymous leaf specializations of the base class *Class* in the UML metamodel.

As illustrated in Figs. 7 and 8, we assume here an extended version of the original definition of OntoUML as proposed in [8]. This extended version of the language also countenances the stereotype «quality»,[7] in order to explicitly represent quality types and in association with a stereotyped relation of «characterization» to represent its ontological counterpart. As discussed in Section 2, the *characterization* relation between an intrinsic trope type and the type it characterizes is mapped at the instance level onto an *inherence* relation between the corresponding individual tropes and their bearers. That means that every instance m of a class M stereotyped as «quality» is existentially dependent on an individual c, which is an instance of the class C related to M via the «characterization» relation. Inherence has the following characteristics: (a) it is a sort of existential dependence relation; (b) it is a binary formal relation; (c) it is a functional relation. These three characteristics impose the following metamodel constraints on the «characterization» construct: by (a) and (c), the association end connected to the characterized type must have the cardinality constraints of one and exactly one; by (a), the association end connected to the characterized type must have the meta-attribute (*isReadOnly* = *true*); «characterization» associations are always binary associations.

Regarding mutability/immutability and necessity/contingency of qualities, we use the following representation strategy. The necessity of a given quality (or, consequently, of a given quality value) is represented by a minimum cardinality $\geqslant 1$ in the association end connected to the «quality» class (or the association end connected to the associated datatype). Alternatively, a contingent quality is represented by a minimum cardinality = 0 in the referred association end. The immutability of a quality (or the corresponding quality value) is represented by using the tagged value *readOnly* applied to the referred association end (i.e., by making its meta-attribute *isReadOnly* = *true*). Finally, the absence of this tagged value in a given association end indicates that a mutable quality (quality value) is being represented.

Finally, in the language, the stereotype «relator» is used to represent the ontological category of relator types. As discussed in Section 2, a relator particular is the actual instantiation of the corresponding relational property. Material relations stand merely for the facts derived from the relator particular and its mediating entities. In other words, relations are logico-linguistic constructions that supervene on relators. Therefore, as argue at length in [8,19], the representation of the relators of material relations must have primacy over the representation of the material relations themselves. In this paper, we simply omit the representation of material relations.

In the sequel, we provide a final example of formal constraints incorporated in the OntoUML metamodel that are derived from its underlying ontological foundations. Relators are existentially dependent entities. Thus, as much as a characterization relation, mediation is also a directed, binary, existential dependence relation. As a consequence, we have that a relation stereotyped as «mediation» in OntoUML must obey the following constraints: (i) the association end connected to the mediated types must have the cardinality constraints of at least one; (ii) the association end connected to the mediated types must have the meta-attribute (isReadOnly = true); (iii) «mediation» associations are always binary associations. Moreover, since a relator is dependent (mediates) on at least two numerically distinct entities, we have the following additional constraint (iv) Let R be a class representing a relator type and let $\{C_1 \ldots C_n\}$ be a set of classes mediated by R (related to R via a *mediation* relation). Finally, let $lower_{Ci}$ be the value of the minimum cardinality constraint of the association end connected to C_i in the mediation relation. Then, $(\sum_{i=1}^{n} lower_{Ci}) \geqslant 2$.

3.1. Discussion

As shown in [8], the distinction among rigid and anti-rigid object types incorporated in the OntoUML language provides for a semantically precise and ontologically well-founded semantics for some of the much discussed but still *ad hoc* distinctions among conceptual modeling constructs. Since its first proposal in this line of work [22], this distinction has had an impact in conceptual model validation [23], in the discovery of important ontological design patterns [24], as well as in the formal and ontological semantics of derived types in conceptual modeling [25]. Moreover, it has influenced the evolution of other conceptual modeling languages, such as ORM 2.0 [26]. Additionally, as demonstrated in [27], this distinction plays an important role also in language engineering, in particular, in the definitions of proper systems of concrete syntax for visual modeling languages. Finally, as argued in [8,28], this distinction has a direct impact even in the choice of different design alternatives in different implementation environments.

Analogously, the explicit representation of intrinsic tropes and quality structures in the language allows for providing an ontological semantics and clear modeling guidelines for attributes, datatypes, weak entities and domain formal relations [19,21,29]. Moreover, the model presented in Fig. 7 illustrates a design pattern for *modeling properties associated to alternative quality structures*. Since its first proposal in [21], this pattern has been applied in several domains (e.g., [30]). Again, from the point of view of language engineering, the trope-based ontology discussed here has been successfully employed in [31,32] in the development of a theoretical framework for model-driven language design.

Furthermore, the strategy for the representation of material relational properties discussed here has been applied in a series of publications to address a number of important and recurrent conceptual modeling problems. For instance, in [19], it was used to address the problem of the collapse of cardinality constraints in material relations; in [33], it has used as an integral part in the development of a solution the so-called problem of transitivity of parthood relations; in [34], in an

[7] Since a class stereotyped as «quality» in this extended version of OntoUML represents a trope type (i.e., a moment type), in an extension of the fragment of the OntoUML metamodel as defined in ([8], p. 334), the homonymous *quality* meta-class should be defined as a specialization of the meta-class *Moment Class*, which, in turn, is a specialization of the base class *Class*.

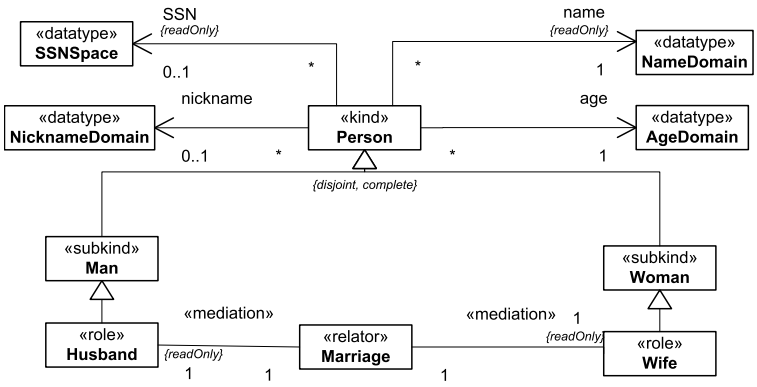

Fig. 9. An OntoUML Conceptual Domain Model with sources of temporal change.

industrial case study of ontology reverse engineering, the systematic identification of the material content of relations (i.e., relators) was reported as a fruitful technique for knowledge elicitation when interacting with domain experts; in [35], the ontological theory of relations underlying this approach has been used to disambiguate the semantics of two fundamental modeling constructs in Goal-Oriented Requirements Engineering; finally, in [36], the same theory has been employed to provide ontological semantics and clear modeling guidelines for disambiguating the constructs of association specialization, association subsetting and association redefinition in UML 2.0.

Because the distinctions and constraints comprising this language are explicitly and declaratively defined in the language metamodel, they can be directly implemented using metamodeling architectures such as the OMG's MOF (Meta Object Facility). Following this strategy, in [37] the authors report on an implementation of an OntoUML graphical editor by employing a number of basic Eclipse-based frameworks such as the ECore (for metamodeling purposes) and MDT (for the purpose of having automatic verification of OCL constraints). An interesting aspect of this strategy is that, once the proper ontological constraints of the underlying foundational ontology have been incorporated in the metamodel, they give rise to syntactical constraints in the language. These constraints in the language metamodel, thus, limit the set of grammatically correct models that can be produced using the language to those whose instances only represent consistent state of affairs according to the underlying ontology.

4. From an ontology-driven conceptual domain model to a computationally-driven specification in OWL

4.1. Temporally changing information in conceptual models

The model of Fig. 9 (termed as *running example* in the remainder of this section) illustrates some important aspects related to change that should be highlighted in the discussion that follows. This model represents a situation in which a person, who can be a man or a woman, is identified by his/her name. Moreover, he/she can have a social security number (ssn) that cannot change. He/she has an age that change annually, and can also be referred by one or more nicknames that may change along his/her life. Finally, a man can get married to only one woman per time (and vice-versa), thus, becoming husband and wife, respectively.

We distinguish here three sources of changes: attributes, relations and type instantiation. Regarding intrinsic properties, we can classify them under the dimensions of **necessary** (mandatory) versus **contingent** (optional), and **mutable** versus **immutable** as discussed in Section 2. Furthermore, the generic dependence between the Kind Person and Quality Type age is also present in the relation *marriedTo* between Husband and Wife (and vice-versa). In other words, both association ends of the relation *marriedTo*, albeit mandatory for the associated types (Husband and Wife), are mutable. Finally, we have a third source of change in this model, related to the anti-rigidity of the role types Husband and Wife. As previously discussed, a particular man instantiates the Role Husband contingently and when mediated by a particular marriage relator. *Mutatis Mutandis*, the same can be said for the role Wife.

4.2. The Web Ontology Language (OWL)

The OWL (Web Ontology Language)[8] is a well known formal language for representing ontologies on the Semantic Web. In this work, we are particularly interested in its DL based variants, to which we refer as OWL in the remainder of this text. DL consists of a family of subsets of classical first order logics that is designed focusing on decidable reasoning. Using DL-based languages, one is able to represent static scenarios with immutable truth-values such that the information about the domain can be completed but cannot be really changed. In particular, the instantiation of a class or property cannot be retracted, except through external intervention. For example, once a model represents that *John being 28 years old* instantiates the class *Husband*, this information cannot be changed.

[8] http://www.w3.org/Submission/OWL/.

DL has two important characteristics to be taken into account here, namely, open world assumption (OWA) and monotonicity. The former entails that what is stated in the model is true but not necessarily the entire truth about a domain, i.e., new information can always be discovered about the represented entities. However, a monotonic logical system is such that the addition of new information/premises must not interfere in the information that has been previously derived. Consequently, what is true in one situation must remain true regardless of any addition of information to the model.

In an OWL model, we can codify the distinction between mandatory versus optional properties (represented by cardinality constraints). However, we cannot represent the distinction between immutable versus mutable, neither the one between rigid versus anti-rigid types. Due to the aforementioned monotonicity of the language, all stated relations, attribute assignments and classification assignments become immutable. In order to circumvent these limitations, a number of authors have been investigating different strategies for representing temporally changing information in OWL [38,39].

Most of these approaches employ a strategy, which consists of interpreting all enduring entities in a domain model (e.g., objects, qualities, relators) as events (processes). This view is grounded in a philosophical stance named *Perdurantism* [40,41]. In a perdurantistic view, a domain individual is seen as a 4D (four-dimensional) "space–time worm" whose temporal parts are slices (snapshots) of the worm. As argued in [6], although such a view can be accurate in representing the current state of knowledge in natural sciences, the distinction between enduring and perduring entities is a fundamental cognitive distinction, present both in human cognition and language. For this reason, as argued in [5], we advocate that such a distinction should be explicitly considered both in conceptual modeling languages as well as in their underlying foundational ontologies. Moreover, besides the philosophical controversy associated with perdurantism, there are a number of issues triggered by such 4D-driven approaches that can become prohibitive for certain design scenarios. Some of these issues are discussed in the next section and are addressed by an alternative approach considered in this article, namely, a reification-driven approach.

Property-reification is definitely not a new idea. In fact, it is a well-known solution for representing temporal information in knowledge representation going back at least to the eighties. Despite this, and despite some clear advantages of this approach for certain design problems, this solution is dismissed in [38] for the lack of an ontological interpretation (or ontological counterpart) for the reified properties. In the next section, we demonstrate that: (i) the ontological categories underlying OntoUML provides for a direct ontological interpretation for these reified entities in the proposed approach; and (ii) these categories can be directly employed for creating transformation patterns between OntoUML models and OWL specifications, in which at least part of the original modal semantics is retained.

4.3. Reifying temporal knowledge in OWL supported by ontological categories

Reification is an operation that makes the reified (objectified) entity amenable to reference, qualification and quantification. In [42], Quine presents reification as a strategy for forging links between sentences or clauses represented in a first order logic (FOL) language. For example, the sentence 'Sebastian walked slowly and aimlessly in Bologna at t' can be reified as $\exists x$ (x is a walk and x is slow and x is aimless and x is in Bologna and x is at t and x is by Sebastian) where x is the objective reference that connect all clauses.

In this section, we are particularly interested in reification as a strategy for representing temporal knowledge using DL-based versions of OWL. It means that we are restricted to a subset of FOL whose predicates are at most binary. For example, the statement 'John is married to Mary at t' is to be reified as something like $\exists x$ (isRelatedTo(x, John) $\wedge$ isRelatedTo(x, Mary) $\wedge$ holds(x, t)). Indeed, in face of this representation some questions arise: what is this thing that is related to *John* and *Mary*? Can this thing keep existing (holding) without being related to both *John* and *Mary*? Are *John* and *Mary* related to each other in the very same way?

In the sequel, we employ the ontological notions defined in Section 2 to answer such questions and to provide ontological meaning for the reified temporal knowledge. More specifically, we intend to reify/objectify the individuals' properties and to attribute to them the time interval during which they hold while having a certain value. For example, the time interval during which John has the age of 27 years old, or the one during which he is married to Mary.

As mentioned, reifying the properties of an individual allows predicating and quantifying over them. It includes attributing to them a time interval during which they are held to be true. Thereby, in Fig. 10, we present two illustrative schemas of applying an ontologically-grounded reification approach to the running example in a temporal view. The object and trope individuals are represent by graphical elements in different shapes, whose projection onto the timeline corresponds to the individual's temporal extension. Moreover, the spatial inclusion of elements represents the *inherence* relation (i.e. the spatially included elements *inhere* in the container) and also reflects the temporal inclusion imposed by the existential dependence. The mandatory properties are represented as rectangles, while the optional properties are represented as rounded corner rectangles. Moreover, the mutable properties are in a lighter grey shade than those immutable ones.

In Fig. 10(a), the larger rectangle represents the object individual *John* that is an instance of the class *Person*; the elements contained in that rectangle represent the qualities corresponding to the reification of John's attributes. Particularly, the quality *John's name* has the same width extension than the individual *John*, representing that it has the same temporal extension of John. In contrast, the necessary and mutable attribute *age* is represented by many qualities (*John's ages*) that together must have the same width extension as the individual *John*. The Fig. 10(b) represents the founding relator of the material relation *marriedTo* between the object individuals *John* and *Mary*, as well as the reification of the correspondent role instantiations (qua individuals). The relator that mediates the couple is represented by the rounded corner rectangle identified as *JMMarriage*, and the qua-individuals that compose it are represented by the elements connected to it by an arrow.

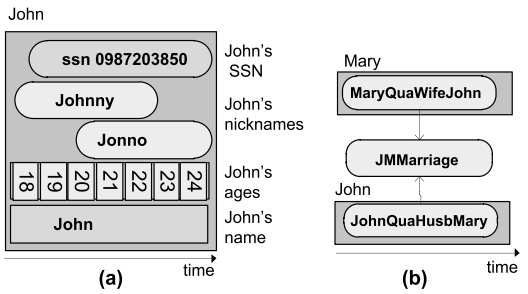

Fig. 10. A schematic representation of an object with (a) its reified qualities; (b) representing reified relators and qua individuals.

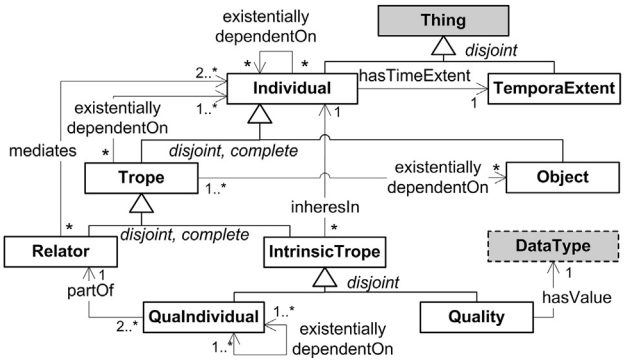

Fig. 11. An ontology-based framework for the systematic reification of properties in OWL. The classes depicted in gray are original OWL constructs that are then specialized by elements of the proposed framework (whose classes are depicted in white).

In Fig. 11, we propose a framework that reflects the ontological notions presented in Section 2 and allows temporal information to be represented in OWL. This framework serves as a *base* (*domain-independent*) *OWL structure* that is to be extended with domain-specific model elements in the manner explained in the sequel.

In this base structure we have that: every individual *has a time extent*; individuals are specialized into *tropes* and *objects*; a *trope* is *existentially dependent* on at least one individual, and can be either a *relator* or an *intrinsic trope*. The former *mediates* two or more individuals, whilst the latter *inheres in* exactly one individual and can be either a *quality* or a *qua-individual*; a *quality* has one *datatype value*; a *qua individual* is *part of* one *relator* and is *existentiallyDependentOn* at least another qua-individual. The relations *inheresIn*, *mediates* and *partOf* are specializations of *existentiallyDependentOn*.

In the remainder of this article, the model of Fig. 9 will be used as support for explaining a number of methodological guidelines discussed in the sequel, which can systematically be used to specialize the base structure represented in Fig. 11:

(a) The rigid/**necessary classes** (e.g. *Person*) should specialize the class ***Object***.
(b) The anti-rigid/**contingent classes** (roles) should be represented as subclasses of the class ***QuaIndividual***. This qua individual type classifies all the qua-individuals resulting from the reification of the participation of individuals of a same object class in a same material relation. For example, the class *Husband* is represented as the class *QuaHusband*, which group all the qua-individuals resulting from the reification of the participation of *Man's* individuals in the material relation *marriedTo*.
(c) The **material relations** of the domain should be explicitly represented as subclasses of class ***Relator***. This relator types classifies all the relator individuals resulting from the reification of the same material relation. For example, the material relation *marriedTo* is represented as the *Marriage* class.
(d) **Attributes** should be represented as subclasses of the class ***Quality***. A quality type classifies all the qualities resulting from the reification of a certain attribute of individuals of the same type. For example, the attribute *name* of the concept *Person* is represented as the class *Name*, which classifies all the quality individuals resulting from the reification of the instantiation of the attribute name of individuals of the class *Person*.

Moreover, we must restrict which and how properties can be or must be applied over the classes. We use the terms minC, maxC and exacC for referring to the minimum, maximum and exact values of cardinality holding for attributes or relation association ends, respectively.

(a) every instance of a **qua-individual** class must ***inheresIn*** exactly one individual of the correspondent object class and only *inheresIn* it. For example, any individual *quaHusband* must *inheresIn* exactly one instance of *Man* and cannot *inheresIn* anything else;

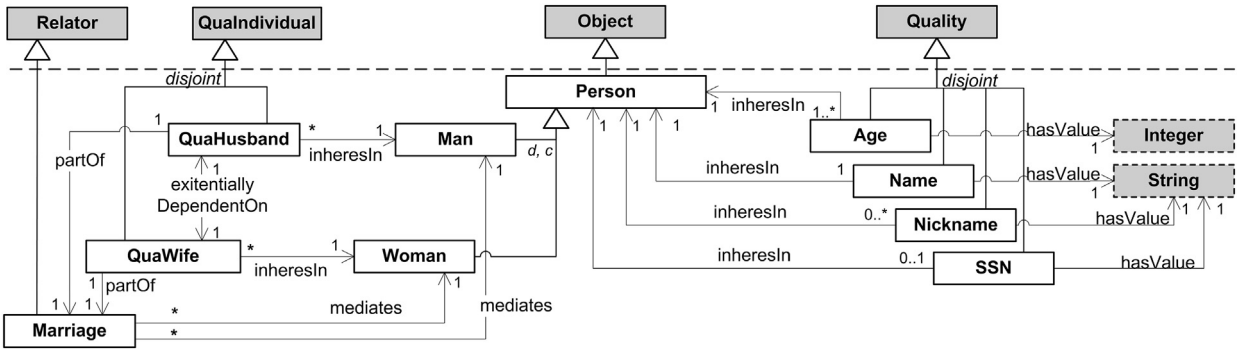

Fig. 12. Mapping the model of Fig. 9 to OWL using the framework of Fig. 11. In this model, the domain independent classes specializing Thing in Fig. 11 (i.e., the classes proposed in our framework representing different ontological categories of individuals) are depicted in full-lined grey boxes; domain-specific classes extending those are represented in full-lined white boxes. Finally, specializations of the OWL construct datatype are represented in dashed grey boxes.

(b) every instance of a **qua-individual** class must be ***partOf*** exactly one individual of the correspondent relator class and only be *partOf* it. For example, any individual *quaHusband* must be *partOf* exactly one instance of *Marriage* and cannot be *partOf* anything else;
(c) every instance of a **qua-individual** class must be ***existentiallyDependentOn*** all other qua-individuals participating in the same relation. For example, any individual *quaHusband* must be ***existentiallyDependentOn*** all other qua-individuals that are part of the relator *Marriage* and cannot be ***existentiallyDependentOn*** any other qua-individual;
(d) every instance of a **relator** class must ***mediate*** only individuals of the correspondent object classes (e.g. an individual of the class *Marriage* must ***mediate*** only instances of the classes *Man* or *Woman*);
(e) every instance of a **relator** class must have as part (***inverse partOf***) only individuals of the qua-individual classes that inhere in the individuals of object classes that the relators mediate (e.g. any individual of the class *Marriage* must have as part only instances of the classes *QuaHusband* or *QuaWife*. These qua individuals inhere in individuals of the classes *Man* and *Woman* mediated by individuals of the class *Marriage*);
(f) every instance of a **relator** class must have as part (***inverse partOf***) at least ***minC***, at most ***maxC*** or exactly ***exactC*** instances of the correspondent qua-individual classes (e.g. any individual of the class *Marriage* must be part of exactly one instance of the class *Man* and exactly one instance of the class *Woman*);
(g) every instance of a **relator** class must ***mediate*** at least ***minC***, at most ***maxC*** or exactly ***exactC*** instances of the correspondent object classes (e.g. any individual of the class *Marriage* must mediate exactly one instance of the class *Man* and exactly one instance of the class *Woman*);
(h) for **immutable material relation**, the domain individuals must be mediated by (***inverse mediates***) at most ***maxC*** or ***exactC*** instances of the **relator** class. Otherwise, if it is **mutable**, **no cardinality restrictions** are imposed to the number of **relators** mediating the domain individuals (***inverse mediates***);
(i) every instance of a **quality** class must ***inheresIn*** exactly one individual of the correspondent object class and only *inheresIn* it. For example, any individual *Name* must *inheresIn* exactly one instance of *Person* and cannot *inheresIn* anything else;
(j) every instance of a **quality** class must ***hasValue*** exactly one value of the correspondent datatype and *only it*. For example, any individual *Name* must *hasValue* exactly one *String* value and cannot be related via *hasValue* to anything else;
(k) for **necessary attributes**, every instance of the correspondent object class must bear (***inverse inheresIn***) at least one instance of the **quality** class. Otherwise, for **contingent attributes**, the minimum cardinality is not restricted. For example, every instance of the class *Person* must have at least one instance of the quality *Age* inhering in it, whilst such restriction does not hold for the quality *SSN*;
(l) for **immutable attributes**, every instance of the correspondent object class must bear (***inverse inheresIn***) at most maxC or exactly exacC instances of the **quality** class. In contrast, for **mutable attributes**, the maximum cardinality is not restricted. It means that every time the attribute changes, a new *quality* individual is necessary for holding the new value. For example, every instance of the class *Person* must have at most one instance of the quality *SSN* inhering in it, whilst such restriction does not hold for the quality *Age*.

Fig. 12 depicts an implementation of the running example following the proposed reification approach. Notice that possible instantiations of this model are the situations illustrated by Figs. 10(a) and 10(b).

4.4. Discussion

In a logical theory representing a conceptual model, time-indexed properties are often represented introducing a temporal parameter in the instantiation relation, i.e. at t, x is an instance of the property P. There are at least three different

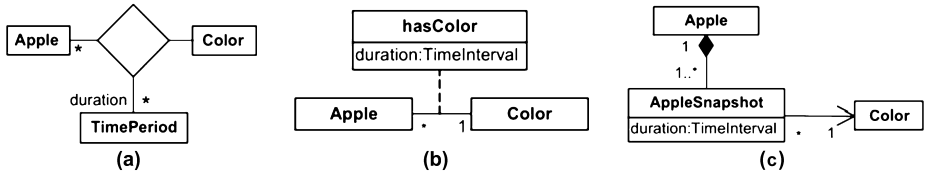

Fig. 13. Different strategies for representing temporally changing information in Conceptual Modeling: (a) time-indexed relations; (b) time modality; (c) temporal snapshots.

interpretations of this temporalization: (i) *t* is just an additional argument that transforms unary properties in binary ones (i.e. in relations) like '*x* is red-at-*t*'; (ii) 'at *t*' is a modal operator that applies to propositions, like 'at *t* (*x* is red)'; (iii) 'at *t*' is a modifier of the particular, i.e., '*x*-at-*t* is red', where '*x*-at-*t*' is, in a four dimensional view, the temporal slice of *x*.

Both option (i) and a solution somewhat similar to option (ii) are widely used in conceptual modeling (see Figs. 13(a) and 13(b)). Option (iii) can be found in some proposals in data modeling (see, for instance, [43]). The view defended here allows for an alternative representation, which is similar but not equivalent to (iii). As previously discussed, this alternative view assumes that a change in an endurant is given by a substitution of tropes, i.e., the temporal information is coded in the temporal extension of tropes. This solution could be easily represented in Fig. 8 without adding complexity to the definition of intrinsic properties. Additionally, this solution has two benefits when compared to (iii). First, as opposed to (iii), one does not necessarily commit to four-dimensionalism, since tropes can be conceived as persisting entities in the same way as substantial individuals (objects). In other words, in the alternative proposed in this paper, one does not have to assume the existence of temporal slices of tropes. Second, in a (Onto)UML class diagram for conceptual modeling, classes are supposed to represent persisting objects such as Apple in (iii), not snapshots of objects such as AppleSnapshot in the same model. Snapshots of objects that instantiate the types depicted in a class diagram are supposed be represented via instance diagrams.

Regarding the conceptual representations in Fig. 13, notice that neither (a) nor (b) could be directly represented in OWL since: (i) OWL cannot represent ternary relations; (ii) OWL properties cannot have properties themselves. Regarding (c), in [39], we have proposed two alternative approaches for representing temporal information in OWL following a 4D (perdurantist) view. In both approaches, we divide the entities in two levels: individual concepts level, for the properties that do not change, and time slice level, for registering the changes on mutable properties. Although that proposal allows one to reasonably represent the intended models, those approaches have the following drawbacks:

- proliferation of time slices: any change occurred in a certain time slice leads to what we call a *proliferation of time slices*. It means that it is necessary to duplicate every time slice in the chain of connected instances that includes the instance on change;
- oddity in ontological interpretation of contingent concepts: in 4D approaches the anti-rigid classes are classes that apply only to time-slices, whilst the rigid classes apply both to 3D entities (ordinary objects, qualities and relators) and their time-slices. This makes the ontological interpretation for the anti-rigid classes (like *Husband* and *Wife*) rather odd;
- repetition of the immutable information on time slice level: the properties that are immutable but not necessary are represented at the time slice level. This leads to a tedious repetition of this information across the time slices of the same individual concept;
- not guaranteeing immutability in the time slice level: since the immutable properties represented at time slice level must be repeated across the time slices of the same individual concept, we cannot guarantee that this property value does not change across time slices.

If we compare the reification approach proposed here with these 4D-based proposals, the following can be stated regarding the aforementioned drawbacks:

- *proliferation of* (*time-slice*) *individuals*: changes no longer cause proliferation of individuals. Although we do have, in this case, the need for new (reified) individuals, the number of these individuals do not increase for each change. For this reason, under this respect, we consider the reification proposal more scalable than the 4D ones;
- *oddity in the ontological interpretation of contingent concepts*: we have homogeneous ontological interpretation for necessary and contingent concepts in the reification proposal;
- *repetition of the immutable contingent information*: except for the mutable properties, no other property is repeated in the reification proposal;
- *not guaranteeing the immutability of contingent properties*: since the immutable properties are represented just once in the reification proposal, its value cannot change.

It is important to highlight that, despite these benefits, there are also limitations and drawbacks in the reification approach. For instance, as pointed out in [38], when reifying relations, we lose the ability to (directly) associate with them meta-properties such as symmetry, reflexivity, transitivity and functionality. However, as discussed in depth in [8], the appli-

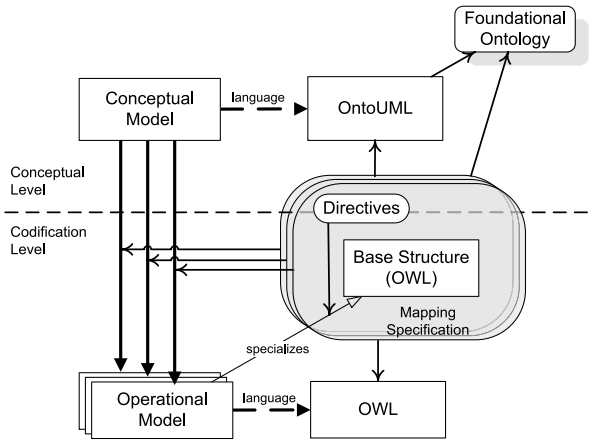

Fig. 14. Using the same Foundational Ontology to systematically support the mappings between conceptual level models (in OntoUML) and possibly multiple codification level models (in OWL).

cation of these meta-properties to material relation is far from a trivial issue. For instance: (i) binary material relations can never just involve the individual being related to itself (since relators must mediate at least two distinct individuals); (ii) symmetry has to differentiate extensional symmetry from intensional symmetry (which can properly be represented here by the roles associated with relators); (iii) transitivity of material relations is an issue of great complexity, which has been partially treated, for example, in [33], for the case of parthood relations.

In any case, this discussion highlights our argument in Section 1 that there is not one single design solution that should fit all design problems. This is by itself enough a good reason for separating conceptual domain modeling from the multiple implementations that can be derived from it and that can be chosen for maximizing specific sets of non-functional requirements.

Finally, although we are aware of initiatives for addressing time representation and reasoning in OWL, we deemed this issue out of scope for this particular paper. However, having a proper axiomatization in that respect is necessary for imposing the temporal restrictions pointed out in our reification proposal, namely: (i) the existential dependence relation must imply temporal inclusion of the dependent individual in the time-extent of the individual(s) it depends on; (ii) a reified necessary and immutable property must have exactly the same time-extent of the individual it depends on; and (iii) a reified necessary and mutable property must have the temporal projection of all its individuals equal to the time-extent of the individual they depend on (i.e. the property age). These issues should be properly dealt with in a fuller approach.

5. Mapping directives and computational support

In this section, we discuss the systematic mapping between OntoUML models and OWL specification using the temporal reification approach discussed in the previous section. As illustrated in Fig. 14, in the conceptual level, we have reference conceptual models represented in a CDML language such as OntoUML. The same OntoUML model can be mapped to different OWL specifications (codification level) in order to meet different design requirements. Each mapping specification makes use of a Base OWL Structure (Fig. 11) and a number of mapping directives. As demonstrated throughout this article, both the conceptual modeling language, on one side, and the mapping directives and base codification structure, on the other, are founded on the same set of ontological categories.

In order to illustrate these mapping directives, in this section, we employ once more the example of Fig. 9. In particular, in what follows, we demonstrate step by step the effects of the mapping directives on the transformation between that OntoUML conceptual model and the resulting OWL specification.

However, before doing that, in Table 1, we present the syntax and semantics of the OWL expressions employed here. The semantics of OWL is defined in a model-theoretic manner such that a (semantic) model in OWL includes a domain $\Delta.^{\mathbf{I}}$ and an interpretation function $.^{\mathbf{I}}$, such that this function maps non-logical constants of the language to individuals in that domain. Analogous, the model-theory of OWL also accounts for a second domain of quantification $\Delta.^{\mathbf{D}}$, whose members are datatype values. Analogously, we have a second interpretation function $.^{\mathbf{D}}$, which maps datatype declarations to values in $\Delta.^{\mathbf{D}}$.

5.1. Kinds and subkinds

Table 1 addresses the mapping directives relative to the ontological categories of kinds and subkinds.

Every instance of Kind has its principle of identity provided by that kind. Since an instance cannot obey different principles of identity [8], all kinds must be disjoint. Moreover, every domain object must have a unique principle of identity

OWL expressions (syntax)	OWL semantics
Declaration(Class(C))	$C^I \subseteq \Delta^I$
Declaration(ObjectProperty(OP))	$OP^I \subseteq \Delta^I \times \Delta^I$
Declaration(Datatype(DT))	$DT^D \subseteq \Delta^D$
SubClassOf(SC C)	$SC^I \subseteq C^I$
EquivalentClasses($C_1 C_2$)	$C_1^I = C_2^I$
DisjointClasses($C_1 \dots C_n$)	$C_j^I \cap C_k^I = \emptyset$, $1 \leqslant j < k \leqslant n$
ObjectUnionOf($C_1 \dots C_n$)	$\{x \mid x \in C_1^I \cup \dots \cup C_n^I\}$
ObjectSomeValuesFrom(OP C)	$\{x \mid \exists y\colon\ (x, y) \in OP^I \wedge y \in C^I\}$
ObjectMinCardinality(n OP C)	$\{x \mid \#\{y \mid (x, y) \in OP^I \wedge y \in C^I\} \geqslant n\}$
ObjectExactCardinality(n OP C)	$\{x \mid \#\{y \mid (x, y) \in OP^I \wedge y \in C^I\} = n\}$
InverseObjectProperty(OP)	$\{(x, y) \mid (y, x) \in OP^I\}$
SubObjectPropertyOf(SOP OP)	$SOP^I \subseteq OP^I$
SubDataPropertyOf(SDP DP)	$SDP^I \subseteq DP^I$
DataSomeValuesFrom(DP DT)	$\{x \mid \exists y\colon\ (x, y) \in DP^I \wedge y \in DT^D\}$

Table 1
Mapping directives for kinds and subkinds.

OntoUML stereotype	Implications to OWL specification
«kind» ***K***	Let $K_1 \dots K_n$ be a number of types stereotyped as *kind*: **DisjointClasses**($K_1 \dots K_n$) **SubClassOf**(K_i Object), for $i = 1$ to n **EquivalentClasses**(Object **ObjectUnionOf**($K_1 \dots K_n$))
«subkind» ***SK***	For each subkind partition P_i, let $SK_1 \dots SK_n$ be a set of specializing subkinds types in the partitions and let S be the specialized rigid type: **SubClassOf**(SK_i S), for $i = 1$ to n If P_i is disjoint: **DisjointClasses**($SK_1 \dots SK_n$) If P_i is complete: **EquivalentClasses**(S **ObjectUnionOf** ($SK_1 \dots SK_n$))

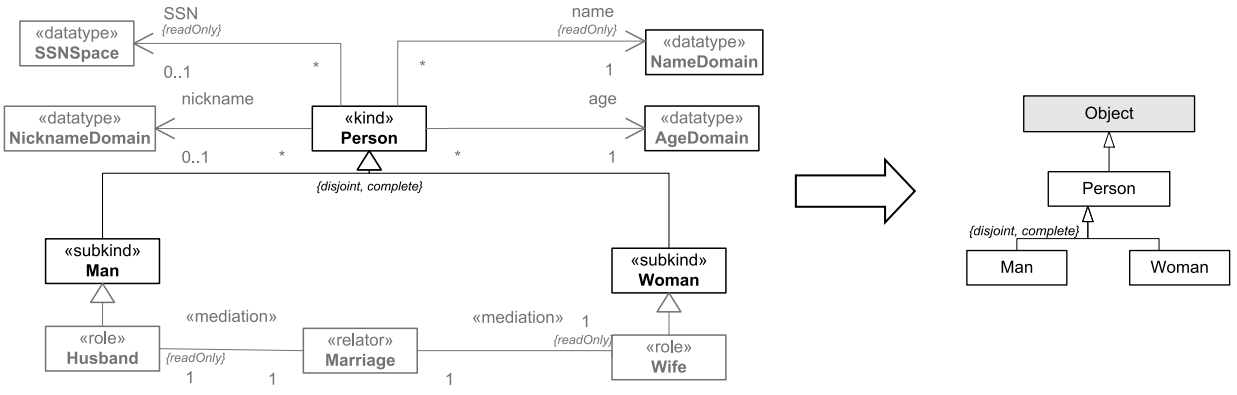

Fig. 15. Mapping of rigid types.

provided by the unique kind it instantiates [8]. Hence, the kinds represented in the model constitute a disjoint and complete partition of the objects of the domain.

Subkinds are types that do not provide a principle of identity for their instances but carry a principle of identity supplied by a unique kind that each of them specialize [8]. In other words, every subkind must specialize (directly or indirectly) exactly one kind (hence the name *subkind*). Like kinds, subkinds are also rigid types, i.e., they must be instantiated by their instances in a necessary manner.

Subkinds always appear in an OntoUML model as a subtype of rigid type (either a kind or another subkind). They frequently appear forming generalization sets (termed subkind partitions), which can either be disjoint, complete, both or none.

In Fig. 15, we illustrate the effect of these mapping directives on the model of Fig. 9.

Table 2
Mapping directives for roles.

OntoUML stereotype	Implications to OWL specification
«role» ***R***	Let $R_1 \dots R_n$ be types stereotyped as *role* in an OntoUML model M: **SubClassOf**(QuaR_i QuaIndividual), for $i = 1$ to n Let SR_i be the direct supertype of R_i in M, **SubClassOf**(QuaR_i **ObjectSomeValuesFrom**(inheresIn SR_i)) {*inheresIn is funcional*} For each role partition P, let $R_1 \dots R_q$ be the subtyping types in the partition and let S be the common supertype: If S is a rigid type: If P is complete: **SubClassOf**(S **ObjectMinCardinality**(1 **InverseObjectProperty**(inheresIn) **ObjectUnionOf** ($\text{QuaR}_1 \dots \text{QuaR}_q$))) Otherwise (if S is anti-rigid, then S is itself reified) **SubClassOf**(QuaR_k **ObjectExactCardinality**(1 existentiallyDependentOf QuaS)), for $k = 1$ tp q If P is complete: **SubClassOf**(QuaS_j **ObjectMinCardinality**(1 invExistentiallyDependentOf **ObjectUnionOf** ($\text{QuaR}_1 \dots \text{QuaR}_q$)))

5.2. Roles

Table 2 addresses the mapping directives for anti-rigid ontological types (*roles*).

Roles are anti-rigid types and represent types that are contingently instantiated by their instances. In the temporal reification approach presented here, roles are represented as reified qua-individuals. Qua individuals inhere in instances of the object types that directly subsume the role type in question. Role types are always specializations of other types (they cannot directly provide a principle of identity for their instances) [8].

Roles can also appear forming generalization sets with other role types (termed role partitions), which can either be disjoint, complete, both or none. For the case of a role partition P, let $R_1 \dots R_q$ be the subtyping roles in the partition and let S be the common rigid supertype. Then we have that:

(i) If the role partition P is complete, then in every instance S there must exist at least one qua individual QuaR_i (corresponding to R_i) inhering in it. Notice that we do not impose here a restriction of at most one qua individual inhering in S for the case that P is disjoint. This is due to the fact that different qua individuals inhering in S can exist distributed in time (e.g., John can bear different John-qua-husband through time).

In the contrary case, i.e., when S is anti-rigid, then we have that:

(i) The type S is itself represented via a type QuaS whose instances are qua-individuals;
(ii) Every qua-individual instance of the reified types QuaRi (corresponding to Ri) must inhere in exactly one qua-individual instance of QuaS;

Finally, if the role partition P is complete, then in every instance QuaS there must exist at least one qua-individual QuaR_i (corresponding to R_i) inhering in it.

In Fig. 16, we illustrate the effect of these mapping directives on the model of Fig. 9. The reader should notice that the cardinality constraint "1..1" in the association end connected to roles (anti-rigid types) in the conceptual model is relaxed to "0..*n*" (∗) in the OWL specification. Again, due to the dynamicity of roles, for example, the same man can bear several different individuals qua-husband through time.

5.3. Relators

Table 3 addresses the mapping directives for *relators*.

Every type stereotyped as relator must specialize (directly or not) the class Relator in the Base OWL Structure. All relator types are rigid types (e.g. a marriage instance cannot cease to be a marriage) [8]. Relators can also appear forming generalization sets with other relator types (termed *relator partitions*), which can either be disjoint, complete, both or none.

In Fig. 17, we illustrate the effect of these mapping directives on the model of Fig. 9.

5.4. The mediation relation

Table 4 addresses the mapping directives for the formal (existential dependence) relation of mediation.

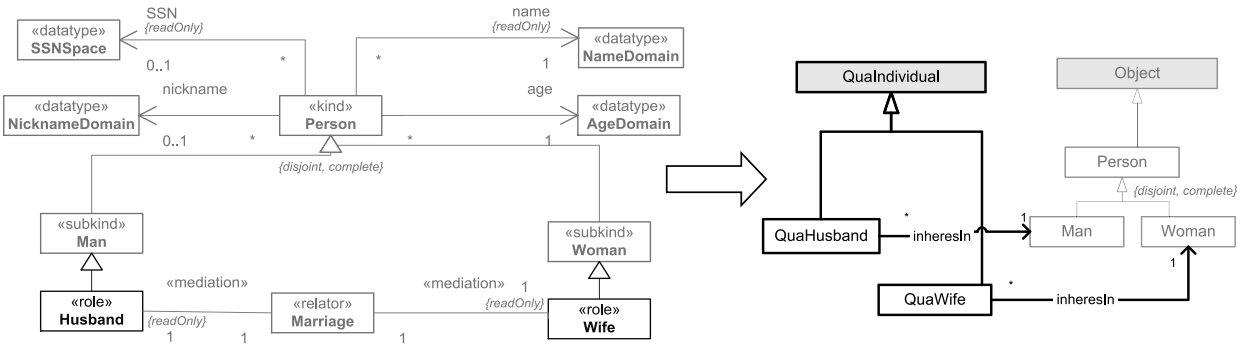

Fig. 16. Mapping roles.

Table 3
Mapping directives for relators.

OntoUML stereotype	Implications to OWL specification
«relator» ***R***	Let R be types stereotyped as *relator* in an OntoUML model M: **SubClassOf**(R Relator) For each relator partition P, let $R_1 \dots R_q$ be the subtyping types in the partition and let SR be the common supertype: **SubClassOf**(R_i SR) for $i = 1$ to q If P is disjoint: **DisjointClasses**($R_1 \dots R_q$) If P is complete: **EquivalentClasses**(SR **ObjectUnionOf**($R_1 \dots R_q$))

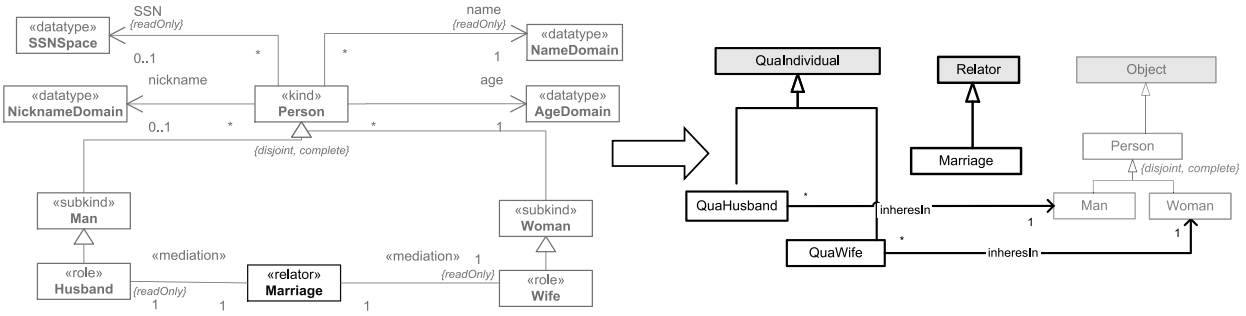

Fig. 17. Mapping relators.

This relation takes place between a subclass of type Relator (R) and other subclass of type Individual ($T_1 \dots T_n$), such that instances of the former mediates instances of the later. The relation of mediation is represented here by the OWL directed binary relation *mediates*.

In the case that ($T_1 \dots T_n$) are all role types then, as previously discussed, they should be represented via their corresponding qua individuals ($\text{QuaT}_1 \dots \text{QuaT}_2$). Since the qua individuals that compose a Relator are mutually existentially dependent on each other, we must represent this existential dependence relation in the resulting codified model. Here, we take the simplified case in which the relator reifies a binary relation, i.e., the relator R mediates two (non-necessarily distinct) roles types T_1 and T_2. In the mapping directives presented here, we represent this mutual dependence relation using the directed binary relation *existentiallyDependentOn* in OWL. This relation is defined to hold between the qua individuals QuaT_1 and QuaT_2 representing the role types among T_1 and T_2.

In Fig. 18, we illustrate the effect of these mapping directives on the model of Fig. 9.

5.5. Intrinsic properties and quality structures

Table 5 addresses the mapping directives for the Intrinsic Properties and Quality Structures.

An intrinsic property is (by definition) owned by exactly one type T. We use here the abbreviations minP and maxP to refer to the minimum and maximum cardinality constraints of that property, respectively. Moreover, we use the meta-

Table 4
Mapping directives for the mediation relation.

OntoUML stereotype	Implications to OWL specification
	For each T_i mediated by R we have that:
«***mediation***»	If *IsReadOnly* = *true* for the association end connected to R and minR $>$ 0 and T_i is a rigid type then: **SubObjectPropertyOf** (mediates ExistentiallyDependentOn)
	If T_i is rigid, If minR $>$ 0 **SubClassOf**(T_i **ObjectMinCardinality**(minR **InverseObjectProperty**(mediates) R)) If maxR $>$ 0 **SubClassOf**(T_i **ObjectMaxCardinality**(maxR **InverseObjectProperty**(mediates) R)) **SubClassOf**(R **ObjectMinCardinality**(minT$_i$ mediates T_i)) {*minT$_i$ is always* > 0} If maxT$_i$ > 0 **SubClassOf**(R **ObjectMaxCardinality**(maxT$_i$ mediates T_i))
	otherwise (if T_i is anti-rigid) **SubClassOf**(QuaT$_i$ **ObjectSomeValuesFrom** (partOf R)) {*part of Relator is functional*} **SubClassOf**(R **ObjectMinCardinality**(minT$_i$ **InverseObjectProperty**(partOf) QuaT$_i$)) {*minT$_i$ is always* > 0} If maxT$_i$ > 0 **SubClassOf**(R **ObjectMaxCardinality**(maxT$_i$ **InverseObjectProperty**(partOf) QuaT$_i$)) Let TRig$_i$ be the most immediate rigid supertype of the anti-rigid types T_i If maxR $>$ 0 **SubClassOf**(TRig$_i$ **ObjectMaxCardinality**(maxR **InverseObjectProperty**(mediates) R)) If minT$_i$ > 0 **SubClassOf**(R **ObjectMinCardinality** (minT$_i$ mediates TRig$_i$)) If maxT$_i$ > 0 **SubClassOf**(R **ObjectMaxCardinality** (maxT$_i$ mediates TRig$_i$))
	Let be the case where relator R is the reification of a binary domain relation such that R mediates the (non-necessarily distinct) role types T_1 and T_2, which in turn are represented by the qua individual types QuaT$_1$ and QuaT$_2$. In this case, we define a *existentiallyDependentOn* relation between QuaT$_1$ and QuaT$_2$ obeying the following constraints: If T_1 is anti-rigid and T_2 is anti-rigid If minT$_1$ > 0 **SubClassOf**(QuaT$_2$ **ObjectMinCardinality**(minT$_1$ externallyDependentOn QuaT$_1$)) If maxT$_1$ > 0 **SubClassOf**(QuaT$_2$ **ObjectMaxCardinality**(maxT$_1$ externallyDependentOn QuaT$_1$)) If minT$_2$ > 0 **SubClassOf**(QuaT$_1$ **ObjectMinCardinality**(minT$_2$ externallyDependentOf QuaT$_2$)) If maxT$_2$ > 0 **SubClassOf**(QuaT$_1$ **ObjectMaxCardinality**(maxT$_2$ externallyDependentOf QuaT$_2$))

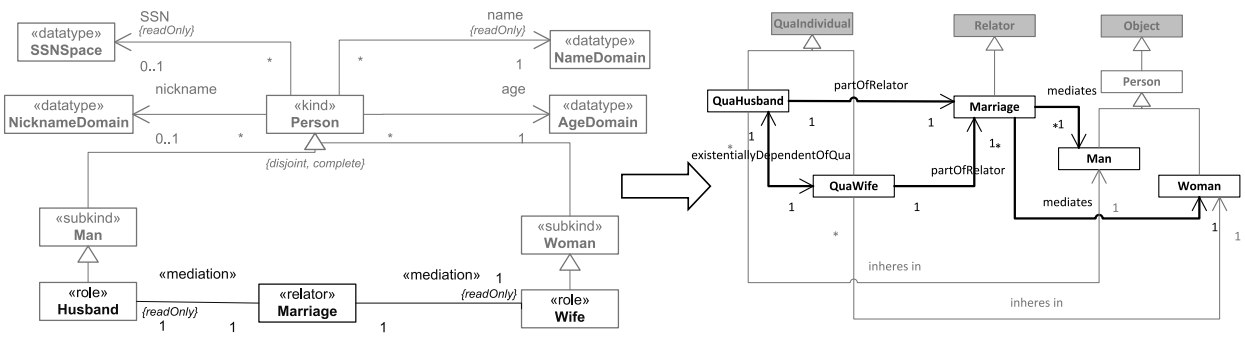

Fig. 18. Mapping relators mediating entities playing roles.

Table 5
Mapping directives for intrinsic properties and quality structures.

OntoUML stereotype	Implications to OWL specification
Intrinsic Property PropT	**SubDataPropertyOf** (PropT Quality) If T is a rigid type T′ = T Otherwise T′ = QuaT, where QuaT is the qua individual associated with T **SubClassOf**(PropT **ObjectSomeValuesFrom** (inheresIn T′)) *{inheresIn is functional}* **SubClassOf**(PropT D**ataSomeValuesFrom** (hasValue *getType*(*PropT*))) *{hasValue is functional}* If minPropT > 0 **SubClassOf**(T′ **ObjectMinCardinality**(minPropT **InverseObjectProperty**(inheresIn) PropT)) If isReadOnly = true (for ProT) and maxPropT > 0 **SubClassOf**(T′ **ObjectMaxCardinality**(maxPropT **InverseObjectProperty**(inheresIn) PropT))

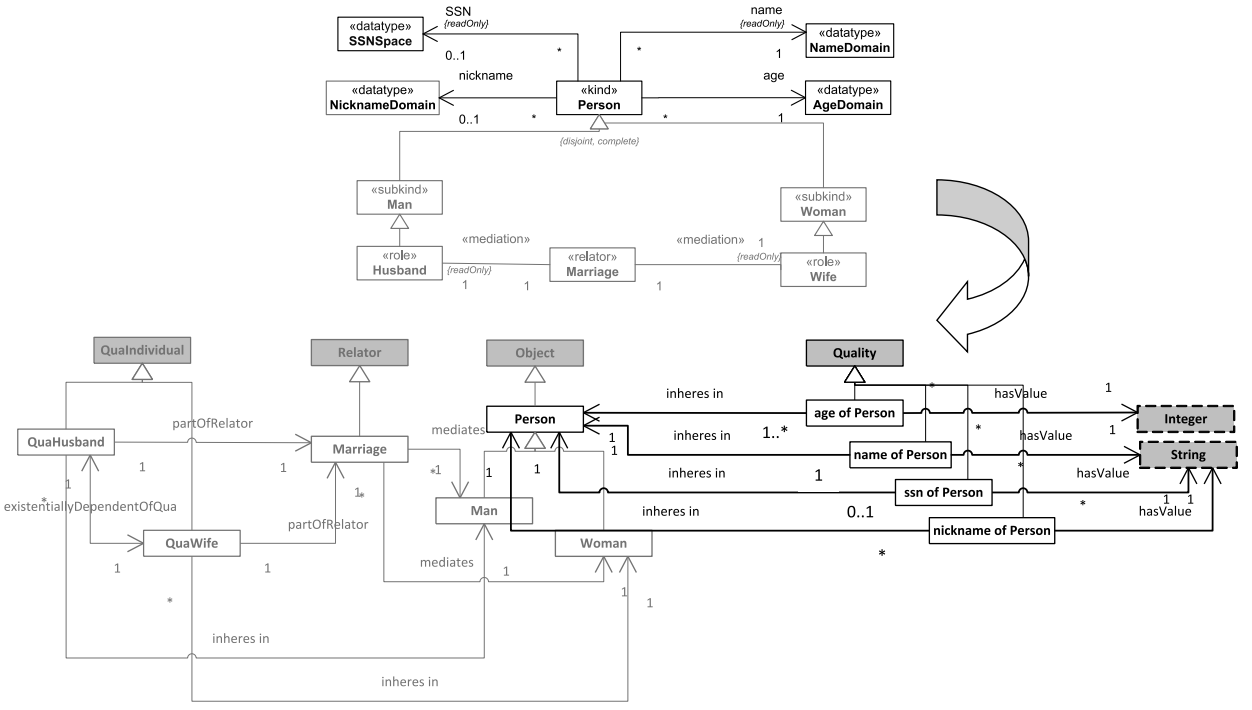

Fig. 19. Mapping intrinsic properties (qualities).

attribute *readOnly* = *true* to represent the *value immutability* of that property. Finally, we use here the function getType to map a property *P* to the datatype associated to *P*.

In the temporal reification approach, a property *P* is not represented directly but reified through a *quality type Q*. For the name of the quality type *Q*, we use the following naming convention here: if we have a quality type reifying property *P* owned by type *T* then this quality type shall be named "*P* of *T*".

The instances of a quality type *Q* inhere in exactly one instance of type *T*. Moreover, each of these quality instances are associated to exactly one value (*hasValue* datatype property) in a datatype *D*.

This datatype *D*, in turn, depends on the type of the property *P* associated to *Q*. In fact, as discussed in depth in [19], OntoUML datatypes can form structures that (albeit richer) are homomorphic to primitive datatypes such as Integer, String, Real, Natural, etc. Given the significant difference in expressivity between OntoUML datatypes and OWL datatypes, in the current version of these mapping directives, we simply chose the OWL datatype with the closest structure to represent each OntoUML datatype.

The number of instances of the quality type *Q* that can be associated to an instance of type *T* depend on the cardinality constraints of the property *P* associated to *Q* as well as the immutability value of PropT in the following manner: if PropT is an immutable property then the maximum cardinality constraint of that property should be respected in the mapping. In contrast, if PropT is mutable then the cardinality constraint should be relaxed to account for property change over time (*trope replacement*).

Finally, in Fig. 19, we illustrate the effect of these mapping directives on the model of Fig. 9.

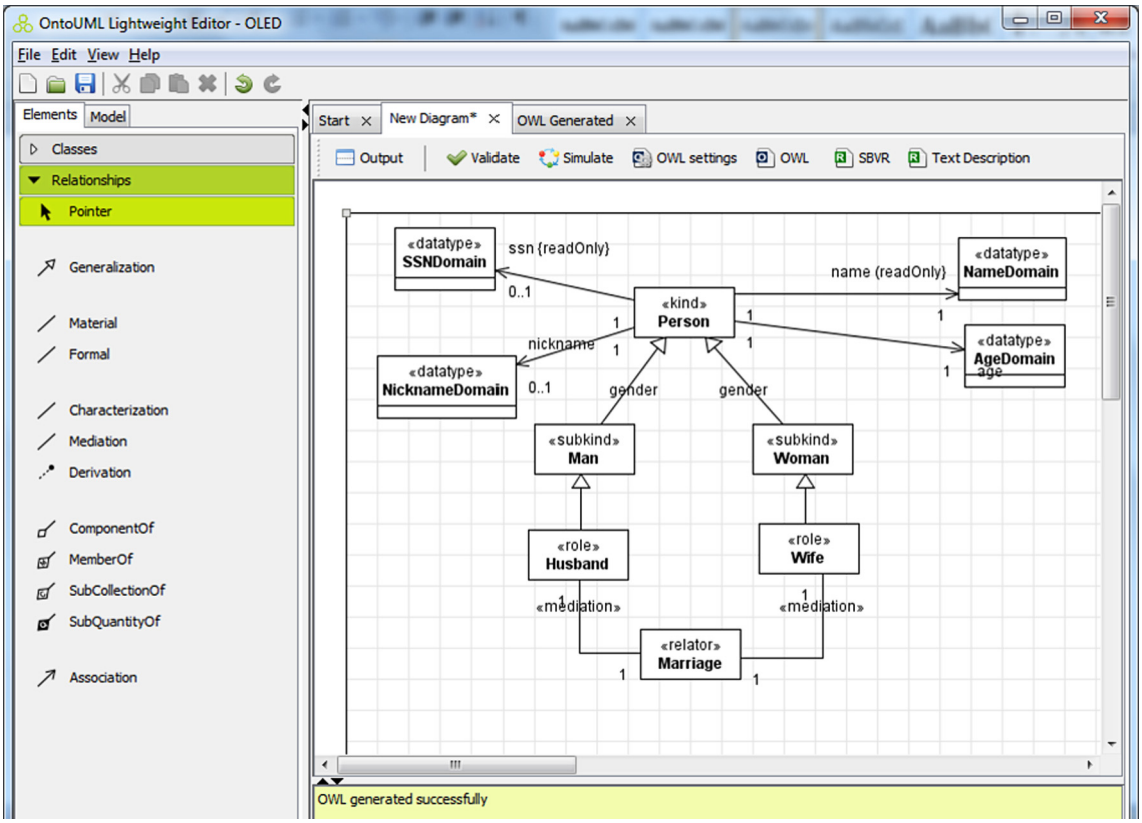

Fig. 20. Example of an OntoUML model built in the OLED Editor.

5.6. Computational support

The mapping directives presented in this section has been fully implemented and incorporated in the Model-Based OntoUML editor discussed in Section 3.1. The OntoUML open-source editor (OLED) can be obtained in https://code.google.com/p/ontouml-lightweight-editor/. The code of the transformation implementing the directives put forth here can be obtained in the same address. Fig. 20 illustrates the model of our running example constructed in the OntoUML editor. Finally, Fig. 21 illustrates the OWL code automatically generated from this OntoUML model by the transformation embedded in the editor.

Once we have stored the information in a reified form, we also need to retrieve them. To illustrate a strategy that can be applied for this purpose, in the sequel, we propose some query patterns using the SQWRL[9] query language. In particular, in these patterns we illustrate how to retrieve information about role instantiations, attribute values and the participation in a material relation. After each pattern, we show a case of its instantiation using the running example of this article.

In these patterns, the underlined classes must be replaced by those regarding the information one wants to retrieve. Furthermore, the two clauses that appear in the head of the query rule refer to the set of variables that will be showed as a result of the query, and to the set of variables that will be taken into account in order to sorting the result, respectively. We assume in these queries that the entity type TemporalExtent (Fig. 11) has two dataProperties, namely startsAt and endsAt, representing start and end points, respectively.

Query Pattern 1. Who has a certain ATTRIBUTE, what is its value and during which time interval does this attribute value hold for that entity?

$$\underline{\textbf{C1}}(?x1) \wedge \underline{\textbf{Quality}_1}(?qlt1) \wedge \textbf{inheresIn}(?qlt1, ?x1) \wedge \textbf{hasvalue}(?qlt, ?v) \wedge$$
$$\textbf{hasTimeExtent}(?qlt1, ?tex) \wedge \textbf{startsAt}(?tex, ?ts) \wedge \textbf{endsAt}(?tex, ?te)$$
$$\rightarrow \textbf{sqwrl:selectDistinct}(?x1, ?qlt1, ?v, ?ts, ?te) \wedge \textbf{sqwrl:orderBy}(?x1, ?qlt1, ?ts)$$

Example. What are the different nicknames that People in my domain have over time?

[9] SQWRL is a query language based on the rule language SWRL (see: http://www.w3.org/Submission/SWRL).

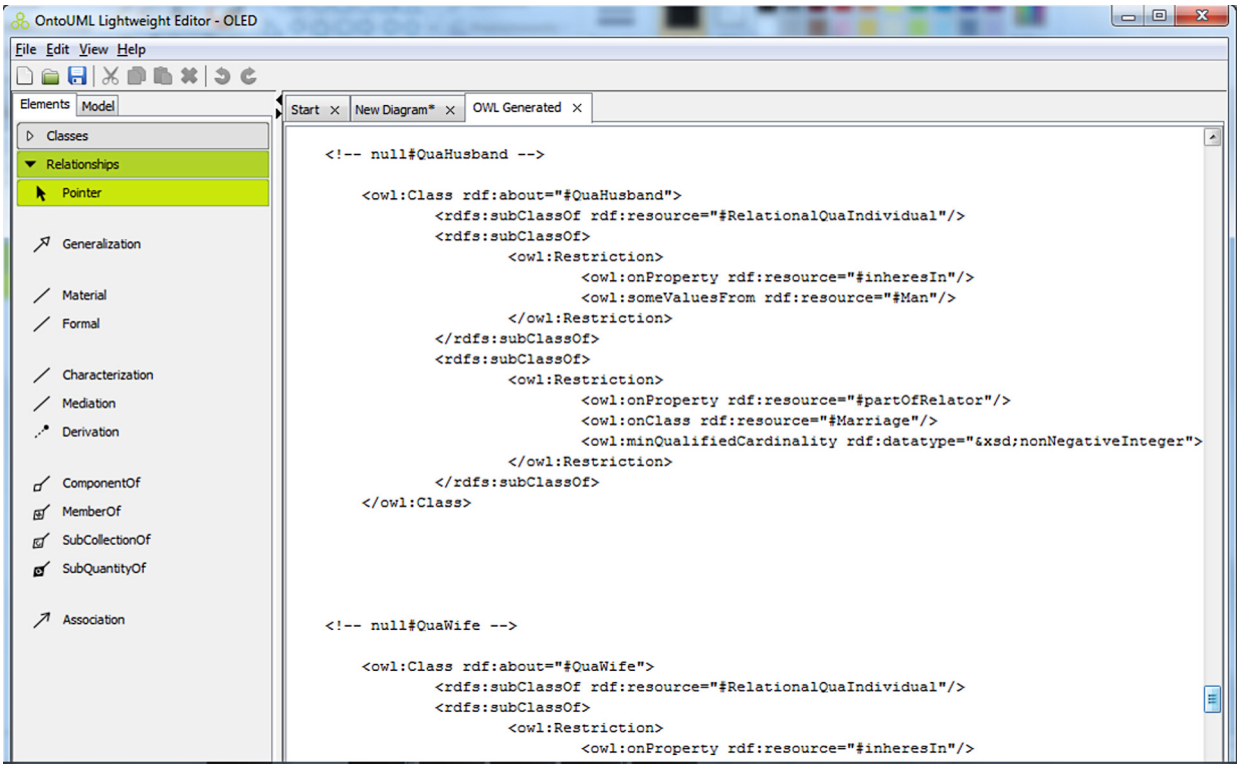

Fig. 21. Automatically generated OWL specification derived from the model in Fig. 19.

Person(?x1) ∧ **nickname**(?qlt1) ∧ **inheresIn**(?qlt1, ?x1) ∧ **hasvalue**(?qlt, ?v) ∧
hasTimeExtent(?qlt1, ?tex) ∧ **startsAt**(?tex, ?ts) ∧ **endsAt**(?tex, ?te)
→ **sqwrl:selectDistinct**(?x1, ?qlt1, ?v, ?ts, ?te) ∧ **sqwrl:orderBy**(?x1, ?qlt1, ?ts)

Query Pattern 2. Who plays a certain ROLE and during which time interval?

C1(?x1) ∧ **Qua**$_1$(?q1) ∧ **inheresIn**(?q1, ?x1) ∧
hasTimeExtent (?q1, ?tex) ∧ **startsAt**(?tex, ?ts) ∧ **endsAt**(?tex, ?te)
→ **sqwrl:selectDistinct**(?x1, ?q1, ?ts, ?te) ∧ **sqwrl:orderBy**(?x1, ?q1, ?ts)

Example. Who have played the husband Role in my domain (i.e., which male individuals have been married before)?

Man(?x1) ∧ **QuaHusband**(?q1) ∧ **inheresIn**(?q1, ?x1)∧
hasTimeExtent (?q1, ?tex) ∧ **startsAt**(?tex, ?ts) ∧ **endsAt**(?tex, ?te)
→ **sqwrl:selectDistinct**(?x1, ?q1, ?ts, ?te) ∧ **sqwrl:orderBy**(?x1, ?q1, ?ts)

Query Pattern 3. Who participates in a certain MATERIAL RELATION playing certain ROLES and during which time interval does this relation hold between these entities (e.g. which people in my domain have been married to each other and in which period)?

C1(?x1) ∧ ··· ∧ **Cn**(?xn) ∧ **Relator**(?r) ∧**Qua**$_1$(?q1) ∧ ··· ∧ **Qua**$_n$(?qn) ∧
inheresIn(?q1, ?x1) ∧ ··· ∧ **inheresIn**(?qn, ?xn) ∧ **partOf**(?q1, ?r) ∧ ··· ∧ **partOf**(?qn, ?r) ∧
hasTimeExtent (?r, ?tex) ∧ **startsAt**(?tex, ?ts) ∧ **endsAt**(?tex, ?te)
→ **sqwrl:selectDistinct**(?x1, ?q1, . . . , ?xn, ?qn, ?ts, ?te) ∧ **sqwrl:orderBy**(?x1, . . . , ?x(n-1), ?ts)

Example. Who participates in a Marriage relation playing the roles Husband and Wife and during which time interval does the relation hold between these entities?

Man(?x1) ∧ **Woman**(?x2) ∧ **Marriage**(?r) ∧ **QuaHusband**(?q1) ∧ **QuaWife**(?q2) ∧
inheresIn(?q1, ?x1) ∧ **inheresIn**(?q2, ?x2) ∧ **partOf**(?q1, ?r) ∧ **partOf**(?q2, ?r) ∧
hasTimeExtent (?r, ?tex) ∧ **startsAt**(?tex, ?ts) ∧ **endsAt**(?tex, ?te)
→ **sqwrl:selectDistinct**(?x1, ?q1,?x2, ?q2, ?ts, ?te) ∧
sqwrl:orderBy(?x1, ?q1,?x2, ?q2, ?ts)

In the way shown here, these query patterns appear verbose and user-unfriendly. However, they do have a very systematic structure and could be easily incorporated in a tool such as the OLED editor such that instantiations of these patterns could be systematically generated.

6. Final considerations

To promote semantic interoperability between information models (and applications that depend on them), we need to be able to guarantee truthfulness to the domain in reality being represented in these models (intra-model consistency). Moreover, we need to guarantee that we establish the correct relations (with the correct semantics) between elements pertaining to different models (inter-model consistency). In order to achieve these objectives, we must rely on representation mechanisms that are able to make explicit their ontological commitments and that are able to capture the subtleties of the subject domains being represented. Moreover, this should be done in a manner that is consistent with how humans as cognitive agents construct and shared their mental models of those subject domains. After all, tasks such as domain understanding, problem-solving and meaning negotiation are meant to be performed by human agents in these scenarios.

Following a tradition on what is now termed *Ontology-Driven Conceptual Modeling*, we argue in this article that these representation mechanisms should be grounded in Foundational Ontologies. In this paper, we present an ontological theory that is built on the idea of tropes (property-instances, moments, modes). This idea affords an ontology that has an illustrious pedigree in philosophy and that has corresponding support both in cognition and language. Moreover, this idea can provide an ontological interpretation and can be used to derive modeling guidelines to many conceptual modeling constructs (e.g., weak entities, reified attributes and associations, datatypes). In this paper, we also present a formal characterization of this theory. As a complement to what has already been presented in [8], the formalization presented here elaborates on different types of characterization relations, systematize the notion of change as trope replacement and includes as part of the logical theory itself (as opposed to present them as formulae schema) the predicates for (anti-)rigidity and for different types of Object types (kind, phased-sortals). This formalization itself contributes to the objective of making available explicit formalizations of these grounding ontological theories in expressive languages such as Common Logic. In particular, it contributes to the effort of developing Common Logic specifications of the categorical core of UFO as presented in [44].

Furthermore, we discuss in this paper how this ontological theory has been employed to engineer an ontologically well-founded conceptual modeling language (OntoUML), which in turn has been successfully employed in a number of complex modeling and semantic interoperability real-world scenarios [28,34]. In the view defended here, in order for a semantic interoperability approach to succeed, we must separate the concerns of conceptual modeling of the domain from the one of conceptual model codification (or implementation). Whilst the former must address issues such as expressivity, clarity, truthfulness to the domain being represented and explicit statement of ontological commitments, the latter must deal with a variety of design demands that are specific to each of the different implementation environments.

In this paper, we also address one specific design problem related to one specific (and important) implementation environment. Despite its success as a computationally efficient logical language for the Semantic Web, OWL renders challenging the representation of temporally changing domain information. However, as discussed [38], this type of information is present in a significant number of real-world scenarios. As one of the main contributions of this paper, we demonstrate how the same trope-theoretical approach that has been used as a foundation for the conceptual domain modeling language OntoUML can be employed to provide a solution to this design problem in a class of description-logics/frame-based languages, represented here by the language OWL.

We put forth here a series of precisely defined directives for systematically mapping OntoUML to OWL using the trope-reification approach proposed for dealing with the design problem aforementioned. We also discuss how these mapping primitives have been implemented in a transformation model and incorporated in a model-based OntoUML editor. Finally, we illustrate how the manipulation of temporal information with the resulting model can be implemented in a semantic web rule/query language such SQWRL.

Acknowledgements

The authors would like to thank Gerd Wagner, Claudio Masolo and Stefano Borgo the fruitful discussions regarding aspects of this work. In particular, we are grateful to Masolo and Borgo for many stimulating discussions regarding the

issue of "change as trope replacement" and, in particular, for important insights regarding the representation of temporal information in conceptual modeling as discussed in Section 4.4.

References

[1] Object Management Group, Semantic Information Model Federation (SIMF): Candidates and Gaps, online, http://www.omgwiki.org/architecture-ecosystem/.
[2] J. Mylopoulos, Conceptual modeling and Telos, in: Conceptual Modeling, Databases and CASE, Wiley, 1992, pp. 49–68 (Chapter 2).
[3] Y. Wand, R. Weber, An Ontological Evaluation of Systems Analysis and Design Methods, Information System Concepts: An In-Depth Analysis, Elsevier Science Publishers B.V., 1989.
[4] J. Recker, M. Rosemann, M. Indulska, P. Green, Do ontological deficiencies in process modeling grammars matter?, MIS Q. Exec. 35 (1) (2011) 57–80.
[5] G. Guizzardi, On Ontology, Ontologies, Conceptualizations, Modeling Languages, and (Meta)Models, Frontiers in Artificial Intelligence and Applications, Databases and Information Systems IV, IOS Press, Amsterdam, 2007.
[6] C. Masolo, S. Borgo, A. Gangemi, N. Guarino, A. Oltramari, Ontology Library, WonderWeb Deliverable D18, 2003.
[7] G. Guizzardi, V. Zamborlini, A common foundational theory for bridging two levels in ontology-driven conceptual modeling, in: 5th International Conference on Software Language Engineering (SLE 2012), Dresden, Germany, in: Lect. Notes Comput. Sci., vol. 7745, 2013, pp. 286–310.
[8] G. Guizzardi, Ontological Foundations for Structural Conceptual Models, Universal Press, The Netherlands, 2005.
[9] E.J. Lowe, The Four Category Ontology, Oxford University Press, 2006.
[10] K. Mulligan, P. Simons, B. Smith, Truth-makers, Philos. Phenomenol. Res. 44 (1984) 287–321.
[11] D. Davidson, The logical form of action sentences, in: Essays on Actions and Events, Oxford University Press, ISBN 9780199246274, 2001.
[12] T. Parsons, Events in the Semantics of English: A Study in Subatomic Semantics, MIT Press, Cambridge, MA, 1990.
[13] M. Fitting, R.L. Mendelsohn, First-Order Modal Logic, Synthese Library Studies in Epistemology Logic, Methodology, and Philosophy of Science, vol. 277, Kluwer Academic Publishers, 1998.
[14] B. Russel, On denoting, Mind 14 (1905) 479–493.
[15] P. Gärdenfors, Conceptual Spaces: the Geometry of Thought, MIT Press, USA, 2000.
[16] B. Heller, H. Herre, Ontological Categories in GOL, Axiomathes, vol. 14, Kluwer Academic Publishers, 2004, pp. 71–90.
[17] K. Mulligan, B. Smith, A relational theory of the act, Topoi 5 (2) (1986) 115–130.
[18] A.L. Thomasson, Fiction and Metaphysics, Cambridge University Press, ISBN 978-0521065214, 1999.
[19] G. Guizzardi, G. Wagner, What's in a relationship: an ontological analysis, in: Proceedings of the 25th International Conference on Conceptual Modeling (ER 2008), Barcelona, Spain, in: Lect. Notes Comput. Sci., vol. 5231, 2008, pp. 83–97.
[20] R.J. Wieringa, W. de Jonge, P.A. Spruit, Using dynamic classes and role classes to model object migration, Theory Pract. Object Syst. 1 (1) (1995) 61–83.
[21] G. Guizzardi, C. Masolo, S. Borgo, In the defense of a trope-based ontology for conceptual modeling: an example with the foundations of attributes, weak entities and datatypes, in: 25th Intl. Conf. on Conceptual Modeling (ER'2006), Arizona, USA, 2006.
[22] G. Guizzardi, G. Wagner, G. Guarino, M. van Sinderen, An ontologically well-founded profile for UML conceptual models, in: 16th International Conference on Advanced Information Systems Engineering (CAISE 2004), Riga.
[23] A.B. Benevides, et al., Validating modal aspects of OntoUML conceptual models using automatically generated visual world structures, J. Univers. Comput. Sci. 16/20 (2010), Special Issue on Evolving Theories of Conceptual Modeling.
[24] G. Guizzardi, A.P. das Graças, R.S.S. Guizzardi, Design patterns and inductive modeling rules to support the construction of ontologically well-founded conceptual models in OntoUML, in: 3rd International Workshop on Ontology-Driven Information Systems (ODISE 2011), London, UK, 2011.
[25] G. Guizzardi, Ontological meta-properties of derived object types, in: 24th International Conference on Advanced Information Systems Engineering (CAISE'12), Gdansk, 2012.
[26] T. Halpin, T. Morgan, Information Modeling and Relational Dababases, Morgan Kaufman, ISBN 1558606726, 2008.
[27] G. Guizzardi, Ontology-based evaluation and design of visual conceptual modeling languages, in: Iris Reinhartz-Berger, Arnon Sturm, Tony Clark, Jorn Bettin, Sholom Cohen (Eds.), Research Directions in Domain Engineering, Springer-Verlag, ISBN 978-3-642-36653-6, 2013, pp. 317–347.
[28] B.T. Bauman, Prying apart semantics and implementation: generating XML schemata directly from ontologically sound conceptual models, in: Balisage Markup Conference, 2009.
[29] A. Albuquerque, G. Guizzardi, An ontological foundation for conceptual modeling datatypes based on semantic reference spaces, in: 7th IEEE International Conference on Research, Challenges in Information Science (RCIS 2013), Paris, ISBN 978-1-4673-2912-5, 2013, pp. 1–12.
[30] S. Fiorini, M. Abel, C. Scherer, A symbol grounding model for the interpretation of 2D-line charts, in: 15th IEEE International Enterprise Computing Conference (EDOC 2010) Workshop Proceedings, Vitória, Brazil, 2010.
[31] B. Henderson-Sellers, Bridging metamodels and ontologies in software engineering, J. Syst. Softw. 84 (2) (2011) 301–313.
[32] B. Henderson-Sellers, On the Mathematics of Modelling, Metamodelling, Ontologies and Modelling Languages, Springer Briefs in Computer Science, Springer-Verlag, Heidelberg, 2012, 106 pp.
[33] G. Guizzardi, The problem of transitivity of part–whole relations in conceptual modeling revisited, in: 21st International Conference on Advanced Information Systems Engineering (CAISE'09), Amsterdam, The Netherlands, 2009.
[34] G. Guizzardi, M. Lopes, F. Baião, R. Falbo, 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. (ISSN 1947-8186) (2010).
[35] R.S.S. Guizzardi, G. Guizzardi, Ontology-based transformation framework from tropos to AORML, in: Social Modeling for Requirements Engineering, in: Cooperative Information Systems Series, MIT Press, Boston, 2010.
[36] D. Costal, C. Goméz, G. Guizzardi, Formal semantics and ontological analysis for understanding subsetting, specialization and redefinition of associations in UML, in: 30th International Conference on Conceptual Modeling (ER 2011), Brussels, Belgium, 2011.
[37] A.B. Benevides, G. Guizzardi, A model-based tool for conceptual modeling and domain ontology engineering in OntoUML, in: 11th International Conference on Enterprise Information Systems (ICEIS), Milan, 2009.
[38] C. Welty, R. Fikes, A reusable ontology for fluents in OWL, in: Proceedings of the 2006 Conference on Formal Ontology in Information Systems: Proceedings of the Fourth International Conference (FOIS 2006), IOS Press, 2006, pp. 226–236.
[39] V. Zamborlini, G. Guizzardi, On the representation of temporally changing information in OWL, in: 15th IEEE International Enterprise Computing Conference (EDOC 2010) Workshop Proceedings, Vitória, Brazil, 2010.
[40] A.C. Varzi, Naming the stages, Dialectica 57 (4) (2003) 387–412.
[41] T. Sider, Fourdimensionalism: An Ontology of Persistence and Time, Oxford University Press, 2003.
[42] W.V. Quine, Events and reification, in: Actions and Events: Perspectives on the Philosophy of Davidson, Blackwell, 1985, pp. 162–171.
[43] M. West, Information Modeling: An Analysis of Uses and Meanings of Associations, PDT Europe, 2002.
[44] G. Guizzardi, G. Wagner, Towards a logic of the ontological dodecagon, in: Pedro Cabalar, Luis Farinas, Agustin Valverde (Eds.), From Philosophy to Computational Logic: Festschrift in Honour of David Pearce's First 60 Years. Special Session at the 13th European Conference on Logics in Artificial Intelligence (IELIA'12), Toulouse, 2012.