Ontology of Dynamic Entities*

Lior Limonad[1,**], Pieter De Leenheer[2,3], Mark Linehan[4], Rick Hull[4], and Roman Vaculín[4]

[1] IBM Haifa Research Lab, Haifa 31905, Israel
liorli@il.ibm.com
[2] Collibra nv/sa, Ransbeekstraat 230, 1120 Brussels 12, Belgium
pieter@collibra.com
[3] VU University Amsterdam, De Boelelaan 1081a, 1081HV Amsterdam, The Netherlands
[4] IBM T.J. Watson Research Lab, Hawthorne, New York, USA
{mlinehan,vaculin,hull}@us.ibm.com

Abstract. This paper describes "dynamic business entities", ontological classes that dynamically acquire and lose properties and relationships as a function of other aspects of the entities. Specifically, we propose how acquisition and loss instructions for such transient properties may be inherent in the definition of the entities possessing them. We use SBVR to demonstrate the specification of dynamic business entities showing how our idea could be applied in practice. We illustrate with an example drawn from the Flanders Research Information Space.

Keywords: Ontology, Conceptual Modeling, Business Entity, Data Integration.

1 Introduction

Enterprise Application Integration (EAI) and *Service Interoperability* are aimed at the realization of cross-party operations through the establishment of software architecture and computer service links between originally independent units. In business, these types of cross-party operations often implement supply channels. A significant practical difficulty among service channel partners is semantic-mismatch due to different understandings of the data communicated among them. The design of such technological ties typically relies on the construction of a *semantic layer*, serving as an underlying conceptualization to help design concrete data and service adapters for overcoming the mismatch. Associated with the business units that take part in the integration, the main product typically being produced at the core of the semantic layer is a corresponding *ontology*, typically expressed using a concrete knowledge representation language (e.g., formal using OWL [12,7] or in controlled natural language expressed in

* The research leading to these results has received funding from the European Community's 7th Framework Programme under grant agreement no. 257593.
** All authors have equally contributed to this work.

P. Atzeni, D. Cheung, and R. Sudha (Eds.): ER 2012, LNCS 7532, pp. 345–358, 2012.

SBVR [11] or visual such as UML class diagrams), providing a shared and formal specification of key business artifacts [6]. Such *business artifacts* reflect mutually agreed upon conceptualized entities that are conceived by the parties as being central to the mutual domain of integration. The better aligned the specification of the semantic layer with the mutual glossary of the partners, the easier it becomes to attain interoperability. Hence, effectively and timely synchronizing data and operations across the enterprise [9,4]. Ensuring such an alignment is anchored in the ontological expressiveness of the language that is used to describe the semantic layer. That is, its capacity to accurately and faithfully describe the mutual domain. As previously theorized in [16,17] and adapted to the purpose of this work, such capacity is inherent in the adherence of the concrete grammar to the following requirements:

1. *Ontological Clarity and Completeness* - refers to the capacity of the grammar to faithfully and comprehensively describe all the business entities and their properties (intrinsic and mutual) as generally perceived by the corresponding community of users. This kind of representation has been traditionally the focus of ontologies, realized by the usage of conceptual modeling such as ER as the specification language to specify entities, attributes, relationships and governing rules.
2. *State tracking* - refers to the capability of the grammar to keep track with domain changes. In the context of the aforementioned business artifact, state tracking entails a unique need to attain *lifecycle-congruency*, which means coherently and unambiguously represent all business entities in different contexts and applications, synchronized with the timely perception of the entities' properties by the partners. For example, it is possible that a manufactured product (e.g., a `car`) is considered as possessing the property `price` only after having it passed all quality assurance tests during its manufacturing. Before that point in its lifetime, the property `price` has no meaning when associated with the `car` being produced.

The main problem with existing grammars that are used for the specification of the semantic layer is their lack of capability to handle lifecycle-congruency. Simply put, there is no existing ontological apparatus that can account for this need in the design grammars that should adequately describe *dynamic entities*. The latter are entities whose possession of some properties (i.e., transient) may be valid at different circumstances within the overall lifetime of the entities.

Therefore, as a solution for the need to account for the external validity of ontological specifications, in this paper, we motivate (in Sect. 2) and propose (in Sect. 3) the conceptual apparatus of an ontology that is designed to handle not only the conceptualization of dynamic entities and the notion of a transient property, but more importantly illustrates the design of a *property possession algebra* for conceptualizing the behavior of transient properties across the lifecycle of corresponding entities.

Next, in Sect. 4, we show how a concrete modeling language (i.e., IBM's Business Artifacts) that is equipped to express business domains that comprise artifacts possessing transient properties can provide the semantic foundation for

a concrete grammar (i.e., SBVR) to improve its lifecycle congruency. The illustration is drawn from a realistic case study in the Flanders Research Information Space, and the implementation is shown in an SBVR-based collaborative ontology management tool, i.e., Collibra's Business Semantics Glossary. Finally, in Sect. 5 we reflect on our work and outline future directions.

2 Motivation and Related Work

In this paper, we introduce the conceptual apparatus that is necessary to precisely describe the dynamics of *transient properties* being part of an *ontology of dynamic entities*. An ontology of dynamic entities is a business ontology whose conceptualized elements are formed by a combination of persistent and transient properties.

1. A *persistent* property is a property whose possession is fixed across the lifecycle of the entity or thing possessing it.
2. A *transient* property is a property whose possession is transient across the lifecycle of the entity or thing possessing it.

Consequently, the specification of the latter type is inherently equipped to determine possession validity at any point in the lifetime of its possessing thing. *Possession* or "expression" is used here to mean that a property is a quality or trait of an entity or a thing. An *entity* is equivalent to the notion of an ontological *kind* (e.g. an SBVR [11] *noun concept* or an OWL *class*), determined by a set of (possessed) *properties* such that the members of a kind are all those and only those things that share all the properties in the given set [3]. Hence, a *thing* means an instance or a member of an entity.

2.1 State of the Art

The goal of this work is to describe all grammar elements being required to specify property *transience* in dynamic entities, while in prior work its has been solely its conceptualization being addressed. For example, previous literature refers to the need for ontological views that are suited for expressing and reasoning about domains that comprise perdurant entities [18] i.e., entities for which their possession of properties and relationships may change in time. Such transient properties have been termed *fluents* in this prior work. Furthermore, a plethora of realization approaches has been accumulated throughout the years, proposing various concrete ways for expressing the existence of transient properties. This prior work includes distinction between optional and mandatory properties in general [5], property negation (i.e., non-possession) [1], the conceptualization of relationships between entities [15], the roles assigned to entities [2], all which may be considered as various forms for the specification of transient properties.

More recently, property transience has been further confirmed and clarified by work in the area of classification [13], similarly distinguishing between base-properties (i.e., persistent) in a class which determine the classification of a thing

(i.e., whether it is a member in a given class), and derived-properties which can be inferred from its membership.

The most fundamental driver justifying the significance of transience is driven by the philosophical paradigm underlying social ontologies according to which there is a clear rational for expressing not only materialized and substantial aspects of the domain, but also aspects being the mere outcome of social intentionality [14]. For example, how would one associate between two individuals (e.g., `John`, and `Kelly`) with the possession of the property `in_love(John,Kelly)` without acknowledging its possession may be transient?

2.2 Novelty of This Work

All of the previous work simply establishes that the existence of property transience is an essence in any linguistic form that is intended to facilitate faithful domain representations. However, none of the previous work is focused on the exact linguistic instrumentation that is necessary to express possession dynamics - i.e., expressing property acquisition and loss in the context of the exact circumstances affecting it such as time, form, association etc. Hence, in this work, our effort is to illustrate the most fundamental machinery that is required to exist in any language that is expected to faithfully and accurately describe possession dynamics. As mentioned above, we find such capability as being most desired in the context of interoperability in which the capability to both describe and interpret entity structures must be synchronized across independent silos.

Note that the possession of a property, whether persistent or transient, may be perceived or viewed by an observer in a way that makes the perception itself transient. We therefore distinguish between the ontological level in which the notion of transience is a fundamental characteristic or trait of the entities being expressed by the ontology regardless of any external view, and the perceptional level in which the perception of property possession may be a function of various contextual and spatial dimensions such as: calendar time, participant's or role's perspective, geography and other. We acknowledge that the perceptional level has been somewhat approached in prior work (e.g., expressed in the form of access controls and views) while the focus of this paper is the ontological level.

Particularly, our proposed solution is designed to account for any case in which possession of properties is a function of phases in the *lifecycle* of the entities being conceptualized by the ontology. Despite the existing body of knowledge aforementioned being focused on various possible conceptualizations for transient properties, existing solutions are essentially different than the one proposed here for either one (or a combination) of the following reasons:

1. Most existing solutions are focused only on a single factor as the potential source for property possession. The most common factor considered is time.
2. Most existing solutions are missing the expressive power needed to explicitly describe how property possession changes as a function of the factor(s) being considered.

Our innovation lies in a solution that:

1. acknowledges the need to associate possession of properties with various factors (e.g., state, geography, perspective). Specifically, we account for situations in which possession should be determined by existential and contextual knowledge about the mere object for which the possession itself has to be determined.
2. suggests the exact processing instructions being required to describe how possession may alternate as a function of the various triggering factors, and specifically as a function of the target objects' lifecycle contexts.

Note, we consider the possession of a property or the attribution of a property to an entity in an inclusive form, uniformly considering the intrinsic traits of entities and also their relationships to other entities as being expressed by properties that one may attribute or predicate about the entity[1]. In case of a relationship, the property is designated by an n-ary predicate being attributed to all entities participating in it (namely, a *mutual property*). This way for example, "having a red color" as an intrinsic property may be attributed to "my car" through the predicate `color(red, my_car)` while associating my car with "myself" as the owner may be expressed as a mutual property through the `owning(self, my_car)` predicate. While the former property in this example is persistent across any point during the lifecycle of `my_car`, the latter may be transient, attributing it to an owner only after the completion of its manufacturing.

Unlike traditional ontologies being merely aimed to express static entities i.e., entities comprising persistent properties only, ontologies of dynamic entities need to express entities comprising combinations of persistent and transient properties. Currently there is no such apparatus. For the latter, the specification also needs to clarify in which circumstances entities' properties may be acquired and dismissed. The dynamics of property acquisition and loss may be expressed as a function of various changes throughout the lifecycle of the entity possessing it. The underlying machinery that is required for such purposes is explained in Sect. 3. In Sect. 4, we demonstrate the advantage of our solution and how the proposed apparatus may be applied to the benefit of business integration and the creation of corresponding shared vocabularies.

3 Description of the Ontology for Dynamic Entities

In this section we describe in detail the internal features of an ontology that is aimed for clarifying the dynamics of property transience in dynamic entities. As mentioned above, in its core, such an ontology distinguishes between the conceptualization of static and dynamic entities and specifically includes a corresponding linguistic capability for describing the behavior that underlie changes

[1] It is worth noting that although the focus in this work is on the dimension of property transience, our findings may be developed further to consider its integration with other previously theorized dimensions of properties (e.g., intrinsic vs. mutual, hereditary vs. emergent etc.)

in the possession of transient properties in dynamic entities. An ontology of dynamic entities should implement the following features: property specifications, possession formula for these properties, and a life cycle context.

Properties of entities are associated with a *possession formula* that is based on values of other properties of entities or can be inferred from lifecycle traces that include the application of a *property possession algebra* i.e., a set of atomic operations about the acquisition and loss of properties, each which may be structured as follows:

- Property specification e.g., `color(car)`, the specification of a property in the ontological description of a domain applies to any property, persistent or transient. In the case of the latter, the specification should also be associated with a possession formula that can be used to evaluate property possession at run-time. Such a formula may be realized as a set of acquisition and loss statements, each comprising two components:
  - A possession instruction i.e., an instruction to either acquire or lose the corresponding property.
  - A lifecycle context (e.g., "when") i.e., a combination of certain property values and certain lifecycle indicators being an antecedent condition to the execution of the possession instruction.

For example, the property `price(1000 €, car)` may be associated with the following possession formula: {`(acquire, on_completion_of_manufacturing)`, `(lose, on_total_loss)`}, i.e., `price` as a property of a `car` is determined as being possessed only after the `car` is fully manufactured. Similarly, in case of extreme damage, a `car` will no longer have a `price`.

It is worth stressing that the aforementioned machinery is stated on a relatively abstract level, keeping it agnostic to possibly more concrete realizations for entity lifecycle styles (e.g., a state-machine), in which the suggested terminology may need to be further specialized. For example, when indeed specified in a form of a state-machine, the concept of possession-formula may be interpreted as being part of functions that describe transition of states. Similarly, other concrete styles may entail different interpretations.

3.1 Designing Possession as a Function of Context

Note that in this example the possession of `price` as a property cannot be expressed intuitively as a function of time: indeed the `on_total_loss` time point is unknown at design time. Hence, from a designer's perspective it is essential (and one may argue also more useful) to provide a grammar (as we illustrate here) that enables formulating the truth of possession as a function of various contextual factors, including factors that stem from inherent information (e.g., materialized) about the possessing entities themselves. As an example, consider the case of the property `attractedTo(matter_1,matter_2)` for which the corresponding possession formula may be formulated as follows:
`{(acquire,opposing(charge(matter_1,value_1),charge(matter_2,value_2))),`
`(lose, NOT opposing(charge(matter1,value_1),charge(matter2,value_2)))}`.

In this case the possibility to determine whether the possession of the property `attractedTo` holds may be inferred directly from the capability to determine the charges `value_1` and `value_2` of both matters.

3.2 Run-Time Evaluation of Possession along the Entity Life Cycle

For convenience purposes, one may use a tagging mechanism at run-time to annotate each property with a possession indicator that is modified each time an acquisition or a loss statement is triggered. This way, instead of needing to evaluate historical traces of acquisition and loss of properties at run-time, there will be an immediate indication for whether the property is possessed or not. Furthermore, in a data-centric approach, lifecycle indicators may themselves be specified as properties such that in the example above, `on_completion_of_manufacturing (true/false, car)` and `on_total_loss(true/false, car)` may both be specified as persistent properties (e.g., as opposed to events). We will illustrate this possibility in SBVR.

Implied from the realization of the above features, the possession of any entity's property can be determined at runtime based on evaluation of the property possession formula. In restricted cases, the possession of a property can be determined by a tool at design time. Given a set of "possession analysis" criteria, this tool can determine (statically) that the expression specified by the possession formula depends upon properties of the entity that have known values at specific stages of the entity lifecycle (i.e., the lifecycle's context). To make this possible, the lifecycle has to be modeled explicitly, specifying notions such as stages and milestones, and the relationships among them as in [10,8]. Static analysis is simplified when lifecycles avoid cycles (i.e. where an entity can return from a later stage to an earlier stage) but is possible in limited circumstances when there are cycles.

Examples given below use the SBVR "Structured English" grammar for convenience. However, the underlying ideas of this paper are agnostic to the concrete grammar that is used to specify an entity lifecycle model. When a property is accessed at some point in the lifecycle, and if the possession formula refers to either (a) persistent properties, or (b) transient properties that are themselves possessed, and if the referenced properties have known values, then a tool can determine that the accessed property is possessed at that point in the lifecycle.

3.3 Mutable Attributes

In addition to the above it is also acknowledged that in current state of the art, a typical realization for the ontological notion of a property is attained through the usage of valued attributes e.g., the persistent property `color(red, car)` may be represented by an attribute `color` being associated with the value "red". Since it is not expected that a `car` will ever change (the value of) its `color` (i.e., a persistent property), once represented as an attribute it may be inferred that the attribute's value is immutable. On the other hand, a transient property such as `owner(person, car)` when represented as a valued attribute may

be determined as mutable, enabling the underlying need to replace the possession of the property with another that indicates a different ownership. Henceforth, the dichotomy of being either mutable or immutable is a direct outcome of using valued attributes as the realization mechanism for properties. Therefore, in the following example, we also indicate for each property whether it is immutable or not.

4 Demonstration

Concluding the feasibility of the apparatus described above, in this section we demonstrate how a concrete modeling grammar (i.e., SBVR) may be used to represent dynamic entities with an example from the Flanders Research Information Space. Finally, by implementing a prototype in the Business Semantics Glossary, we show how the design of ontology for dynamic entities and the creation of shared vocabularies and rules in general may benefit each other.

4.1 Flanders Research Information Space

The Web is a catalyst for *open innovation*. Enterprises and research institutions have come to realize that they no longer can rely on their own research to innovate, but instead share or trade ideas and results to achieve a greater benefit to themselves and others. The Flemish government has taken the lead at driving European open innovation through Flanders Research Information Space (FRIS[2]), an ambitious change program that publishes data on innovation-related entities such as research institutes, researchers, and funded projects.

Many of these FRIS entities share the characteristics of dynamic entities. E.g., take a research project: they usually have long life cycles (up to several years), requiring FRIS data relating to properties (i.e., publications, deliverables, consortium) of these entities to be updated regularly. Secondly, not all properties are intended for publication (e.g., periodic review reports). Thirdly, the data related to these properties have to be provided by different parties (such as principle investigator, consortium members, project officer) and according to a certain semantics (in the case of FRIS based on the Common European Research Information Standard (CERIF[3]). Finally, they exhibit transient properties: e.g., the start date of a project is only valid if the project has been formally initiated. Summarising: an ontology for the dynamic entity Project should declare and enforce: "what are the attributes of an entity in which stage of the entity's lifecycle?" Currently CERIF is formalized using the ER grammar, which does not allow to model transient properties or possession formula.

4.2 Two Associated Dynamic Entities: Proposals and Projects

We illustrate our approach in terms of two FRIS entities that acquire or lose possession of properties and relationships in function of other aspects of the

[2] `http://www.researchportal.be`
[3] `http://www.eurocris.org/`

entities. We follow the SBVR practice of underscoring nouns (SBVR noun concepts), showing relationships (SBVR verb concepts or fact types) using italics, and using bold face for keywords such as "if". This approach does not require any change to SBVR, which already provides for conditional necessity rules.

Consider an entity Proposal with the following properties (in terms of SBVR binary verb concepts).

1. Persistent + immutable:
   Proposal *isownedby* Principle Investigator / Principle Investigator *owns* Proposal
2. Persistent + mutable:
   Proposal *isdescribedby* Discipline Code/ Discipline Code *describes* Proposal
3. Transient + immutable:
   Proposal *has* Evaluation Score / Evaluation Score *of* Proposal
4. Transient + mutable:
   Proposal *defines* Work Plan / Work Plan *isdefinedby* Proposal

Consider an entity Project with the following properties (in terms of SBVR binary verb concepts):

1. Persistent + immutable: Project *executes* Proposal / Proposal *is executed by* Project
2. Transient + mutable: Project *has* Start Date / Start Date *of* Project

For the persistent properties of Proposal and Project we define integrity constraints that are true independent of the stage in which the entity is:

1. **It is necessary that each** Proposal *isownedby* **exactly one** Principle Investigator
2. **It is necessary that each** Proposal *isdescribedby* **at least one** Discipline Code
3. **It is necessary that each** Project *executes* **exactly one** Proposal

Now for the transient properties we define the possession formula in terms of the following assumed lifecycle stages.

- For Proposal: → Submitting → Evaluating → Notifying → Submitting
- For Project: Initating → Reviewing → Finishing

For our example, we define "milestones" (related to achieving the end of these stages) as special types of characteristic in SBVR. A characteristic is a unary verb concept with Boolean type. We have four milestone characteristic types for Proposal:

1. Proposal *has been submitted*
2. Proposal *has been evaluated*
3. Proposal *is accepted*
4. Proposal *is rejected*

We have three "milestone" characteristic types for Project:

1. Project *is initiated*
2. Project *has been reviewed*
3. Project *is finished*

We can express for every transient property P of an entity E a (dis-)possession formula that has to be true if one or more characteristics M_i is/are true. A possession formula may use an SBVR "necessity" modality:

– **It is necessary that each** Proposal *defines* **exactly one** Work Plan **if the** Proposal *has been submitted*.

A dispossession formula may use an SBVR "impossibility" statement:

– **It is impossible that a** Proposal *defines* Work Plan **if the** Proposal *has* **not** *been submitted*.

A combination possession formula for entity-property Proposal has Work Plan using "if and only if" allows a shorthand notation for the conjunction of the two previous "if" statements. This version uses "always" as an alternative way to express "necessity":

– **A** Proposal **always** *defines* **exactly one** Work Plan **if and only if the** Proposal *has been submitted*.

The following are some example combination formulae. Combination possession formula for Proposal *has* Evaluation Score:

– **It is necessary that each** Proposal *has* **at least one** Evaluation Score **if and only if the** Proposal *has been submitted* **and the** Proposal *has been evaluated*.

Combination possession formula for Proposal *is executed by* Project:

– **It is necessary that each** Proposal *is executed by* **exactly one** Project **if and only if the** Proposal *is submitted* **and the** Proposal *has been evaluated* **and the** Proposal *is accepted* **and the** Proposal *is* **not** *rejected* **and the** Project *is initiated*.

Combination possession formula for Project *has* Start Date:

– **Each** Project **always** *has* **exactly one** Start Date **if and only if the** Project *is initiated* **and the** Proposal *that is executed by* Project *is accepted*.

The two above possession formula illustrate that the possession of properties may also depend on aspects of another entity. E.g., in the latter example, the validity of property Project *has* Start Date depends on an aspect of another entity Proposal that is associated (through the verb concept "*executed by*'), this aspect being the milestone: Proposal *is accepted*.

4.3 Transitivity of Possession

The above formula may be simplified if we define production rules for each sequence of two milestones. E.g., "Proposal *has been evaluated*" implies "Proposal *has been submitted*". This also assumes that Proposal *is accepted* and Proposal *is rejected* are mutually exclusive, the latter which can be specified with SBVR's mutual exclusion contraints.

Therefore, from the following (repeated from above):

- **It is necessary that each** Proposal *is executed by* **exactly one** Project **if and only if the** Proposal *is submitted* **and the** Proposal *has been evaluated* **and the** Proposal *is accepted* **and the** Proposal *is* **not** *rejected* **and the** Project *is initiated*.

We can infer automatically:

- **It is necessary that each** Proposal *executed by* **exactly one** Project **if and only if the** Proposal *is accepted* **and the** Project *is initiated*.

4.4 Acyclic Lifecycles

As defined above, part of the lifecycle of a Proposal has no cycles, meaning that a Proposal cannot "go back" from "*has been accepted*" to "*is rejected*". Thus a tool is able to statically determine that some transient properties of Project are available once the "Proposal *has been accepted*" and "Project *has been initiated*".

4.5 Implementation in Business Semantics Glossary

Figure 1 shows a screenshot of the noun concept Proposal within the "Proposal" vocabulary managed by the "CERIF" speech community that is part of the "FRIS" semantic community. The Business Semantics Glossary is a tool that implements the *business semantics management* (BSM) methodology [4]; hence for collaboratively managing the semantics of persistent properties of CERIF entities. The screenshot shows the attributes we defined in Subsect. 4.2 for the dynamic entity Proposal: fact types to express properties, characteristics denoting milestones, integrity constraints for persistent properties, and possession formula for transient properties. The underscores define hyperlinks to other parts in the glossary showing the embedding of our approach in the broader context of creating shared vocabularies based on CERIF for service interoperation in FRIS.

The right hand panel shows the governance settings: in the bottom-right corner is indicated which member in the community (here "Pieter De Leenheer") carries the role of "steward", who bears final accountability. The status "candidate" indicates that the term is not yet fully articulated: in this case 37.5%. This percentage is automatically calculated based on the articulation tasks that have to be performed according to the BSM methodology. Tasks are related to defining attributes and are distributed among stakeholders and orchestrated using workflows.

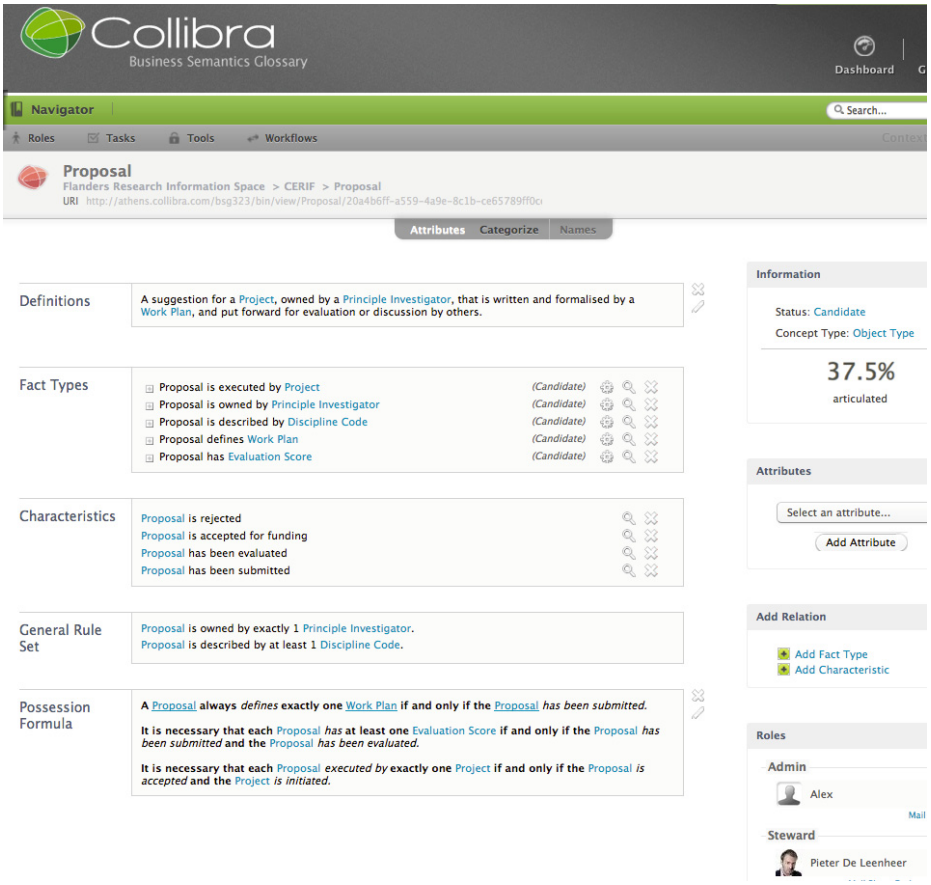

Fig. 1. Collaborative designing dynamic entities in the Business Semantics Glossary

5 Discussion and Future Work

The fundamental contribution of this work is prescribing the essence of any language that is aimed to adequately describe dynamic entities. Particularly this includes the unique mechanism that is needed to handle transience through possession and loss instructions being inherent in the language. Currently, no commercial tool, aside from the prototype in Collibra's Business Semantics Glossary used to demonstrate the feasibility of our solution, exists. Hence, we find our effort in this work as paving the road towards the development of improved data-integration tools that are better equipped to facilitate inter-silos communication. This may include tools such as IBM Infosphere MDM, Oracle Master Data Management, and SAP Enterprise Master Data Management.

In addition, our contribution has similar applicability to existing knowledge representation standards (e.g., OWL) and software development languages which

may be extended with the proposed capabilities as well. Preliminary penetration of the solution is starting to show its first signs in technologies such as Java JSR-305, enabling basic annotation of "nullable" properties. Yet, more expressive solutions as proposed here seem to further alleviate the need to mitigate both the burden in handling reference availability mismatch at runtime, and also enabling richer static analysis of code.

In order to accommodate for the exact features expected in the underlying tools to adequately reason about property transience and inference about possession validity, our most immediate intention is devoted towards further investigation of specification well-formedness and discovery of inconsistencies.

Aside from possible applications of the proposed solution, future research work may be aimed to extend the proposed linguistic machinery with the capability to express epistemic aspects of property possession. This may include the possibility for example to attach roles to every possession formula indicating who is responsible for what milestone. Such capability is essential in data governance.

When business partners exchange information in Enterprise Application Integration (EAI) and Service Interoperation scenarios, they should distinguish persistent versus transient attributes of that information. Traditional modeling and ontology standards do not enable these distinctions. The method we propose exploits existing capabilities of SBVR to explicitly identify which attributes are valid under what circumstances. This removes doubt about exchanged information, and should improve the success of interoperation scenarios.

References

1. Allen, G.A., March, S.T.: The Proper Role of Optionality and Negation in Conceptual Modeling. In: Proceedings of the Eighth Annual Symposium on Research in Systems Analysis and Design, Richmond, VA, May 21-23 (2009)
2. Bera, P., Burton-Jones, A., Wand, Y.: The effect of domain familiarity on modelling roles: an empirical study. In: Proceedings of PACIS 2009, p. 110 (2009)
3. Bunge, M.: Treatise on basic philosoph. In: Ontology I: The Furniture of the World, vol. 3. Reidel, Boston (1977)
4. De Leenheer, P., Christiaens, S., Meersman, R.: Business semantics management: a case study for competency-centric HRM. Computers in Industry 61(8), 760–775 (2010)
5. Gemino, A., Wand, Y.: Complexity and clarity in conceptual modeling: Comparison of mandatory and optional properties. Data and Knowledge Engineering 55(3), 301–326 (2005)
6. Gruber, T.: A translation approach to portable ontology specifications. Knowledge Acquisition 5(2), 199–220 (1993)
7. Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P.F., Rudolph, S. (eds.): OWL 2 Web Ontology Language: Primer. W3C Recommendation (October 27, 2009), http://www.w3.org/TR/owl2-primer/
8. Hull, R., Damaggio, E., Fournier, F., Gupta, M., Heath III, F(T.), Hobson, S., Linehan, M., Maradugu, S., Nigam, A., Sukaviriya, P., Vaculin, R.: Introducing the Guard-Stage-Milestone Approach for Specifying Business Entity Lifecycles (Invited Talk). In: Bravetti, M. (ed.) WS-FM 2010. LNCS, vol. 6551, pp. 1–24. Springer, Heidelberg (2011)

9. de Moor, A., De Leenheer, P., Meersman, R.: DOGMA-MESS: A Meaning Evolution Support System for Interorganizational Ontology Engineering. In: Schärfe, H., Hitzler, P., Øhrstrøm, P. (eds.) ICCS 2006. LNCS (LNAI), vol. 4068, pp. 189–202. Springer, Heidelberg (2006)
10. Nigam, A., Caswell, N.S.: Business artifacts: An approach to operational specification. IBM Systems Journal 42(3), 428–445 (2003)
11. OMG: Semantics of Business Vocabulary and Business Rules (2008), http://www.omg.org/spec/SBVR
12. OWL Working Group: OWL 2 Web Ontology Language: Document Overview. W3C Recommendation (October 27, 2009), http://www.w3.org/TR/owl2-overview/
13. Parsons, J., Wand, Y.: Using cognitive principles to guide classification in information systems modeling. MIS Quarterly 32(4), 839–868 (2008)
14. Searle, J.R.: Social ontology. Anthropological Theory 6(1), 12–29 (2006)
15. Wand, Y., Storey, V.C., Weber, R.: An ontological analysis of the relationship construct in conceptual modeling. ACM Trans. Database Syst. 24(4), 494–528 (1999)
16. Wand, Y., Weber, R.: On the ontological expressiveness of information systems analysis and design grammars. Information Systems Journal 3(4), 217–237 (1993)
17. Wand, Y., Weber, R.: On the deep structure of information systems. Information Systems Journal 5(3), 203–223 (1995)
18. Welty, C., Fikes, R.: A Reusable Ontology for Fluents in OWL. In: Proceedings of FOIS, pp. 226–236. IOS Press (2006)