# Implementing Organizational Self Awareness
## *A Semantic Mediawiki based Enterprise Ontology Management Approach*

David Aveiro[1,2,3] and Duarte Pinto[1]
[1]*Exact Sciences and Engineering Centre, University of Madeira, Caminho da Penteada 9020-105 Funchal, Portugal*
[2]*Madeira Interactive Technologies Institute, Caminho da Penteada, 9020-105 Funchal, Portugal*
[3]*Center for Organizational Design and Engineering, INESC-INOV Rua Alves Redol 9, 1000-029 Lisboa, Portugal*

Keywords: Enterprise Engineering, Model, Meta-Model, Abstract Syntax, Concrete Syntax, Wiki, Semantic Web, DEMO.

Abstract: In this paper we present a solution currently being developed to enable collaborative enterprise ontology model management using the Semantic MediaWiki as a base tool. This solution is solidly grounded on the theoretical foundations of Organizational Self-Awareness and φ-theory of enterprise ontology and is a valuable contribution to facilitate general and distributed enterprise model management and also concrete and abstract syntax specification, i.e., the specification of a language's meta-model. This allows flexibility and ease of use in creation and adaptation of organizational models and also the use of semantic queries as to detect and inform users on any violation of meta-model rules.

## 1 INTRODUCTION

A large amount of time is lost, in organizations, in the handling of unknown exceptions causing dysfunctions as exception handling can sometimes take almost half of the total working time, and the handling of, and recovering from, exceptions is expensive (Saastamoinen and White, 1995). On another hand, current Enterprise Engineering (EE) approaches seem to lack in concepts and method for a continuous update of organizational models, so that they are always up to date and available as a more useful input for the process of continuous change of organizational reality and decision on possible evolution choices. It seems that the root problem is an absence of concepts and method for explicit capture, and management of information of exceptions and their handling, which includes the design and operationalization of organization artifacts (OA) – e.g., actor role pizza deliverer – that solve caused dysfunctions. Not immediately capturing this handling and the consequent resulting changes in reality and the model of reality itself, will result that, as time passes, the organization will be less aware of itself than it should be, when facing the need of future change due to other unexpected exceptions. The lack of awareness of organizational reality has been addressed with the coining of the term "Organizational Self-Awareness" (OSA), presented and refined in (Magalhaes et al., 2007) and (Zacarias et al., 2007). OSA stresses the importance and need of continuously available, coherent, updated and updateable models of organizational reality. With our research work we aim to facilitate distributed awareness of organizational reality and also coordinated distributed change of models of the enterprise's reality using adequate methods and software tools as a support. In our tool development efforts a necessity arose of allowing a precise way of conceptualizing and implementing the separation of several concerns, while keeping coherence and integration between them, namely: organizational reality; models of reality; and their representations, while also having adequate flexibility for the specification of model and meta-model evolution.

## 2 RELATED WORK

We ground our research in a particular Organizational Engineering approach, namely, the Design & Engineering Methodology for Organizations (DEMO) (Dietz, 2006). Our research – presented in this and the next sections – are heavily based in DEMO so, while proceeding, the

Aveiro D. and Pinto D..
Implementing Organizational Self Awareness - A Semantic Mediawiki based Enterprise Ontology Management Approach.
DOI: 10.5220/0004550404530461
In *Proceedings of the International Conference on Knowledge Engineering and Ontology Development* (SSEO-2013), pages 453-461
ISBN: 978-989-8565-81-5

reader which is unfamiliar with this methodology is advised to also consult (Dietz, 2006) or (Dietz and Albani, 2005) or other publications in: www.demo.nl. From several approaches to support EE being proposed, DEMO seems to be one of the most coherent, comprehensive, consistent and concise (Dietz, 2006). It has shown to be useful in a number of applications, from small to large scale organizations – see, for example, (Dietz and Albani 2005) and (Op' t Land, 2008) (p. 39). Nevertheless, DEMO suffers from the shortcoming referred in the introduction. Namely, DEMO models have been mostly used to devise blueprints to serve as instruments for discussion of broader scale organizational change or development/change of IT systems (Op' t Land, 2008) (p. 58) and does not, yet, provide modeling constructs and a method for a continuous update of its models as reality changes. Current software tools supporting DEMO also suffer from the same shortcoming.

Having in mind the needs of: (1) generalized access and awareness of organizational reality; and (2) facilitating incremental and coherent changes to models of organizational reality, as well as meta-model evolution and model migration; we are developing an EE tool based on the Semantic MediaWiki software. This tool intends to facilitate communication of organizational models, online editing and change of models while also providing dynamic diagram generation in the Wiki pages representing an organization. In the next sections we present other related work that serves as a theoretical foundation for the contributions presented in this paper.

## 2.1 Basic Ontological Notions

We adopt the ontological system definition from (Dietz 2008) (citing (Bunge, 1979)) which concerns the construction and operation of a system. The corresponding type of model is the white-box model, which is a direct conceptualization of the ontological system definition presented next. Something is a system if and only if it has the next properties: (1) composition: a set of elements of some category (physical, biological, social, chemical etc.); (2) environment: a set of elements of the same category, where the composition and the environment are disjoint; (3) structure: a set of influencing bonds among the elements in the composition and between these and the elements in the environment; (4) production: the elements in the composition produce services that are delivered to the elements in the environment. From (Dietz, 2008) we find that in the

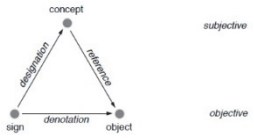



Figure 1: The meaning triangle.

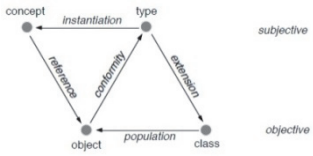



Figure 2: The ontological parallelogram.

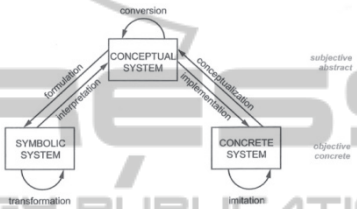



Figure 3: The model triangle.

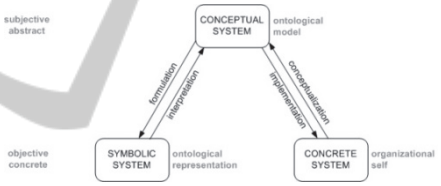



Figure 4: Model triangle applied to organizations.

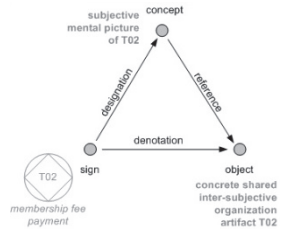



Figure 5: Meaning triangle applied to a transaction OA.

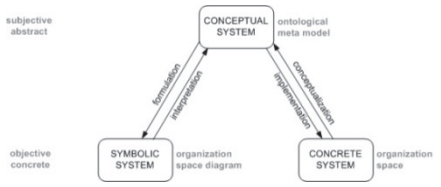



Figure 6: Model triangle applied to the organization space.

Ψ-theory based DEMO methodology, four aspect models of the complete ontological model of an organization are distinguished. The Construction

Model (CM) specifies the construction of the organization: the actor roles in the composition and the environment, as well as the transaction kinds in which they are involved. The Process Model (PM) specifies the state space and the transition space of the coordination world. The State Model (SM) specifies the state space and the transition space of the production world. The Action Model (AM) consists of the action rules that serve as guidelines for the actor roles in the composition of the organization.

In Figures 1 and 2, respectively, the meaning triangle and the ontological parallelogram, taken from (Dietz, 2005) which explain how (individual) concepts are created in the human mind. We will also base our claims in the model triangle, taken from (Dietz, 2006) and presented in Figure 3. We find that the model triangle coherently overlaps the meaning triangle. This happens because a set of symbols – like a set of DEMO representations (signs) that constitute a symbolic system – allows the interpretation of a set of concepts – like a set of DEMO aspect models, part of the ontological model, constituting a conceptual system. This conceptual system, in turn, consists in the conceptualization of the "real" inter-subjective organizational self, i.e., the set of OAs constituting the concrete organization system's composition structure and production. Figure 4 is an adaptation from the model triangle of Figure 3 and depicts our reasoning. We call the set of all DEMO diagrams, tables and lists used to formulate the ontological model as ontological representation.

Now relating with the meaning triangle, we can verify that a particular sign (e.g., a transaction symbol with label membership fee payment), part of an ontological representation (e.g., actor transaction diagram, representing a library's construction model) designates (i.e., allows the interpretation or is the formulation) of the respective concept of the particular transaction part of the respective ontological model (e.g., construction model). This subjective concept, in turn, refers to a concrete object of the shared inter-subjective reality of the organization's human agents (e.g., the particular OA transaction T02). Figure 5, an adaptation from the meaning triangle depicts this other reasoning.

Another example of an OA related with T02 would be the transaction initiation OA, relating T02 with actor role registrar (also designated by A02) and formulated by a line connecting the transaction and actor role symbols of T02 and A02. Actor role registrar is, in turn, another OA of the construction space of the library. Once such role is communicated to all employees of a library, it becomes a "living" abstract object part of the shared inter-subjective reality of the library's human agents. Such object, along with other OAs of the organizational inter-subjective reality, give human agents a way to conceptualize their organizational responsibilities – in this case, requesting membership fee payments to aspirant members. We name this set of all abstract objects living in the inter-subjective reality of an organization's members as the organizational self. By explicitly formalizing this set of abstract objects that we call *organization artifacts* and making this formalization and their representation available and changeable in a distributed way we aim to achieve organizational self-awareness, described in more detail in (Aveiro and Pinto, 2013).

## 2.2 EE Tools Supporting DEMO

To generalize the access and awareness of the organizational reality is not a trivial task. Such tool must not only enable the collection of distributed and coherent organizational knowledge aligned with the organizational reality but also be understandable and of easy use by any of the organization's collaborators. This collection of organizational knowledge should be in an integrated repository of both the conceptual understanding and the symbolic understanding in the form of diagrams and tables. There are some solutions for DEMO modeling like Visio (Microsoft 2010) (only diagrams), Xemod (MPRISE 2010) and ModelWorld (Hommes, 2013), but we found ourselves facing the same issues with all of them. For our objectives Visio would be the less helpful, as it offers no support for anything but diagram specification, and even that support is achieved by custom made stencils that until now have a very limited way of enforcing the rules and/or restrictions of the modeling language. Visio also fails to help us with our needs of generalized access and awareness of organizational reality and facilitating incremental changes to models of organizational reality, as it does not present a way of offering a generalized access to the knowledge nor it facilitates any kind of coherent incremental change. A change in a diagram is exclusively a change in that diagram, it does not propagate to other diagrams that share the same organizational fact.

Xemod is a tool built exclusively for DEMO modeling and as such offers another level of support. This support comes at a cost, as this is also a far more expensive tool than a basic license of Visio. But even though in Xemod we have a set of rules to help us model and support for the whole

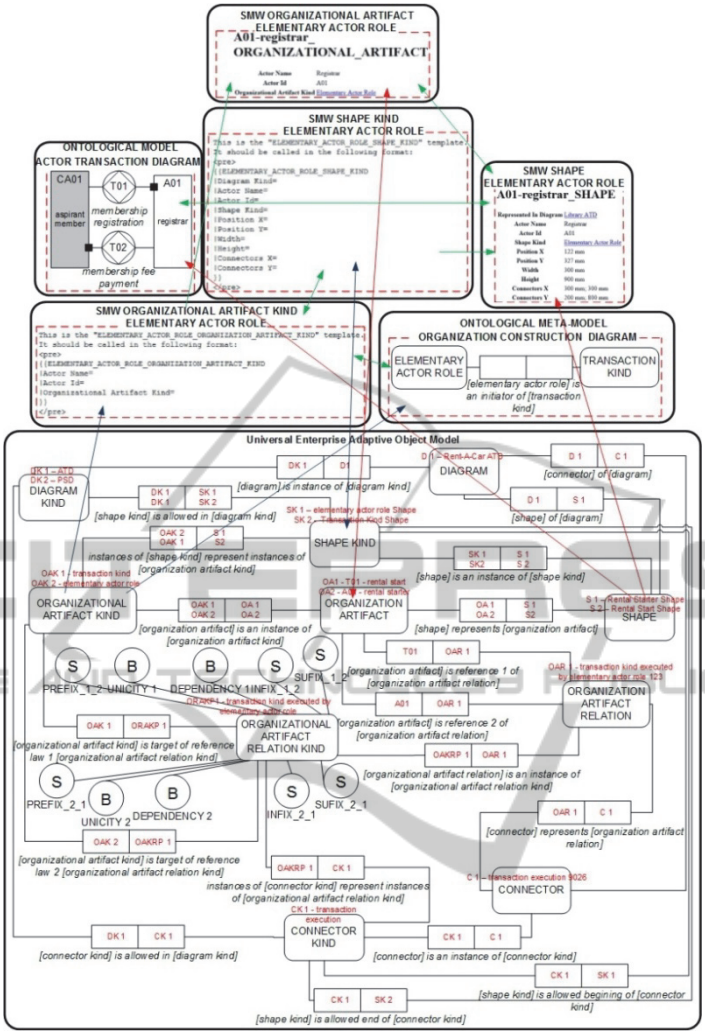



Figure 7: Overview of the SMW Pages and their Relation with DEMO Models and UEAOM.

methodology and not only the diagrams, Xemod also has its issues: (1) it is impossible to change the pre-existing stencils, so if the standards change, you are most likely having to pay once more to upgrade your tool to the newer version; (2) ineffective way offered to propagate knowledge – unless you have Xemod installed on every workstation, the way of sharing your organizational knowledge is by exporting it to a far more complex Access database or Excel spreadsheet; (3) we find another flaw that is shared with Visio, although in Xemod there is some sort of change propagation, as your changes in a Transaction Result Table reflect in all diagrams, it still doesn't provide a full support in the changes as, for instance, changing an Object Fact Diagram class name, does not reflect neither in the name of related fact types nor in the name of related result type name.

ModelWorld is an online modelling and diagramming tool for business architecture models and, unlike the previous two, is free. As it runs on the browser it has the advantages of being platform independent and allowing for collaborative modelling and validations. Just like in Xemod, ModelWorld has a set of rules on how diagram shapes in DEMO modelling relate, but in the same way it does not allow for the stencils change either, leaving you at the mercy of the updates, in this case however, free of charge. But all three tools fail to meet the needs; in facilitating the incremental and integrated changes. ModelWorld too offers very little support in this important aspect and, for a generalized access and awareness, like Visio, ModelWorld only allows to export the information in diagram form.

These problems encountered in current state-of-

the-art tools led us to developing our own tool as to fully support and implement our notion of organizational self-awareness and the evolution of the meta-models governing such awareness.

# 3 REALIZING OSA WITH A WIKI

The Wikipedia article on Berlin article contains many links to other articles, such as «Germany» and «European Union». However, the link to «Germany» has a special meaning: it was put there since Berlin is the capital of Germany. To make this knowledge available to computer programs, one would like to «tag» the link [[Germany]] in the article text, identifying it as a link that describes a «capital property». With Semantic MediaWiki (SMW), this is done by putting a «property name» and «::» in front of the link inside the brackets, thus: [[Is capital of::Germany]]. In the article, this text still is displayed as a simple hyperlink to «Germany». The additional text «capital of» is the name of the property that classifies the link to Germany. Information that was provided in an article is now provided in a formal way accessible to software tools. We foresaw that SWM could be an adequate base to develop an EE tool supporting and facilitating OSA. For that we took some implementation decisions described next.

## 3.1 Fundamental Patterns Used

As a theoretical base for the realization of OSA with a SMW we have specified in (Aveiro and Pinto 2013) the Universal Enterprise Adaptive Object Model (UEAOM). This model is represented in a diagram in the World Ontology Specification Language (Dietz, 2005), a derivative of the Object Role Modelling (ORM) language (Halpin, 1998). Due to the inherent preciseness and first order logic predicate behind ORM, also WOSL is a very adequate language for our goal to specify in a powerful and precise way all aspects of models, the respective meta-models, their representations as well as their evolution. The classes of our UEAOM follow the *type square* and the *adaptive object model* patterns (Yoder et al., 2001), usually applied to software engineering to allow dynamic and runtime evolution of a software system's services, but here applied to the enterprise engineering context, precisely to allow dynamic and runtime evolution of not only organization systems, but also of the meta-models governing the structure and instantiation of the elements of organization systems. Following Figure 7, certain wiki pages will be the specification of objects that are instances of the following classes of our UEAOM: ORGANIZATION ARTIFACT KIND (OAK) – for the specification of meta-model elements; SHAPE KIND – whose pages specify shape kinds whose instances will represent instances of OAKs; ORGANIZATION ARTIFACT – whose pages specify OAs – at model level – that are instances of certain OAKs; and SHAPE, whose pages specify particular shapes – at diagram level – representing particular OAs. A similar reasoning is followed for the classes ORGANIZATION ARTIFACT RELATION KIND (OARK) – relation kinds between OAKs; CONNECTOR KIND; ORGANIZATION ARTIFACT RELATION; and CONNECTOR. An example of a SHAPE that represents – at diagram level – an OA (itself, in turn, at model level) consists in the page titled: «A01-rental_starter_shape». A page called «A01-rental_starter» will be an OA represented by the before-mentioned shape. This OA, in turn, is an instance of the OAK, itself specified by the page «ACTOR ROLE». This page, in turn relates to page «ACTOR ROLE SHAPE KIND» specifying the characteristics of a shape to represent the OAK specified by the page «ACTOR ROLE». As we can see by these examples, one of the advantages of using the adaptive object model (AOM) and type square patterns is that they allow a systematic and precisely organized instantiation at several levels and concerns, namely: AOM classes level, meta-model, model and representation concerns – while keeping all relevant relationships between objects.

On each page one has to specify semantic wiki properties to provide semantics to the respective object of the UEAOM, relating it with all other relevant objects. In the example, by adding to the page «A01-rental_starter» the link: [[is_instance _of::ACTOR ROLE]], we specify with the property *is instance of* that A01 is an instance of the meta-model level OAK specified by the page «ACTOR ROLE». All pages that are instances of OAKs or OARKs need to specify the property [[is_instance_of::ORGANIZATION ARTIFACT KIND]] or [[is_instance_of::ORGANIZATION ARTIFACT RELATION KIND]] respectively. Following this method we can, for example, use the SMW mechanism of semantic queries and automatically obtain a list of all OAs, OARs and also the respective SHAPEs and CONNECTORs that represent them. We can also execute queries to dynamically obtain the current version of the meta-model, specified by the pages instances of OAKs and OARKs.

## 3.2 Creating Models and Diagrams

Our implementation based in a SMW serves the purpose of not only formally specifying the meta-model behind models and diagrams, but also of the organizational self and its change and to visualize and edit diagrams automatically and dynamically generated from the pages and their semantic properties. Each ORGANIZATION ARTIFACT page makes no sense by themselves, and need to be contextualized in a user friendly way. This contextualization is achieved in two steps, the first is to establish relations between the OAs by creating ORGANIZATION ARTIFACT RELATIONS and the second is representing such OAs in a DIAGRAM. Just like OAs and OARs are instances of certain OAKs and OARKs, DIAGRAMS will be instances of a certain DIAGRAM KIND. A page specifying a DIAGRAM KIND, in turn allows the formalization of which SHAPE KINDs are allowed in a certain DIAGRAM. For example the «ACTOR TRANSACTION DIAGRAM KIND» page specifies that only the presence of SHAPE KINDs «ACTOR ROLE SHAPE» and «TRANSACTION KIND SHAPE» is allowed in instances of this diagram kind. Concrete diagrams of an organization like, for example, the EU-Rent Actor Transaction Diagram (ATD), are pages specifying instances of the UEAOM class DIAGRAM and these pages, in turn, have to contain the property *is_instance_of_diagram_kind*::ACTOR TRANSACTION DIAGRAM KIND. These wiki pages that specify concrete diagrams have a special behavior implemented by an extension to SMW. These pages output a DIAGRAM generated in run time environment using those OA pages and their semantic properties, automatically generating an image implemented in the Scalable Vector Graphics (SVG) format: an open format allowing easy import/export operations and also zoom operation without losing resolution quality. The page «A01-rental_starter_shape» is an example of an instance of a SHAPE and the page «CA02-Driver.is_the_executor_of.T03-car_drop-off_connector» is an example of an instance of the UEAOM class CONNECTOR, and semantically associated with the page «transaction_execution_connector_kind», itself an instance of CONNECTOR KIND. At meta-model level, instances of classes SHAPE KIND and CONNECTOR KIND will be associated with instances of classes SHAPE PROPERTY like, for example, «actor_id» and of CONNECTOR PROPERTY like, for example, «line_color». At model level these properties are instantiated as objects instances of classes SHAPE PROPERTY VALUE and CONNECTOR PROPERTY VALUE. For example, «A01» and «Black», respectively. These are more examples of the application of the type square pattern, also applied in the case of OAKs and OAKRs furthermore showing the immense power to our approach. Instances of properties and values could have been also implemented as wiki pages but the most appropriate approach was to use the mechanism of semantic properties already present in SMW.

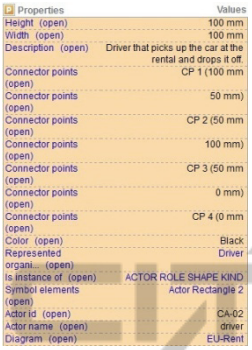
Thus, classes of our UEAOM that include the name property are usually implemented as properties in the respective pages and classes including the term VALUE, as values of the semantic properties themselves in the respective wiki pages. The dynamic power of type square power is kept as we can dynamically change properties that can be associated with kinds by editing the special wiki pages of templates.

To facilitate the process of creation of models and their representations, and also make the changes directly made in SVG diagrams reflect in SMW pages and their properties, a java script based diagram editor is currently under development to implements all the functionality deemed convenient like ones present in well known modeling tools such as Microsoft Visio.

## 3.3 Page Names and Semantic Properties

A standard specification is an explicit set of requirements for an item, material, component, system or service. The need to define a standard nomenclature for wiki pages is crucial to create a homogeneous model and ensure compatibility with other projects that may be developed and integrated with this. A wiki page representing a ORGANIZATION ARTIFACT KIND or ORGANIZATION ARTIFACT RELATION KIND at meta-model level consists of capital letters and words are separated by underscore. For example, the wiki page for representing a «transaction kind» fact type should be «TRANSACTION_KIND». A wiki page representing an instance of a ORGANIZATION ARTIFACT is a little different. It consists of a capital letter followed by an order number, and then a hyphen, followed by the name of the fact, where words are lower cased and separated by underscore. For example, the wiki page «A01-rental_starter» is defined by the capital letter «A» (standing for ELEMENTARY ACTOR ROLE) followed by the number «01», and then a hyphen

followed by the actor name “rental_starter”, both of these are ORGANIZATION ARTIFACT PROPERTY VALUES, used here together to form an identifier of the page. On another example, the wiki page «CA02-driver» is defined by the capital letters «CA» (standing for COMPOSITE ACTOR ROLE) followed by the number «02» and then a hyphen followed by the composite actor name “driver”.



| Properties | Values |
|---|---|
| Height (open) | 100 mm |
| Width (open) | 100 mm |
| Description (open) | Driver that picks up the car at the rental and drops it off. |
| Connector points (open) | CP 1 (100 mm |
| Connector points (open) | 50 mm) |
| Connector points (open) | CP 2 (50 mm |
| Connector points (open) | 100 mm) |
| Connector points (open) | CP 3 (50 mm |
| Connector points (open) | 0 mm) |
| Connector points (open) | CP 4 (0 mm |
| Color (open) | Black |
| Represented organi... (open) | Driver |
| Is instance of (open) | ACTOR ROLE SHAPE KIND |
| Symbol elements (open) | Actor Rectangle 2 |
| Actor id (open) | CA-02 |
| Actor name (open) | driver |
| Diagram (open) | EU-Rent |

Figure 8: Semantic box for CA02-driver_shape.

Properties also have a simple standard nomenclature. Any property consists of lowercase words separated by underscore. Examples of valid properties are: «actor_id», and «initiating_actor_role». Every class represented in the UEAOM will have their own set of properties that need to be implemented (given values) for the creation of instances of such class. There is no typical SMW page for the classes of our UEAOM, these are specified in the implementation as templates and forms that we will explain in greater detail in section 3.4. As an example, Figure 8 depicts the semantic box that shows the properties for the wiki page «CA02-driver_shape», an instantiation of the class SHAPE.

The instances of OARK and OAR are a particular case when it comes to the nomenclature. Here we are dealing with composed OAKs and OAs with multiple elements, and as such this has to be considered in the naming. For example in an Actor Transaction Diagram we have two kinds of OARKs, the «ACTOR_ROLE.is_an_initiator_of. TRANSACTION_KIND» and «ACTOR_ROLE.is_ the_executor_of.TRANSACTION_KIND». As previously, and maintaining the coherence, at the meta-model level the OARKs consist of capital letters separated by underscore, but here composed with possible a prefix, an infix and/or a suffix, in lower case, also separated by underscore to create the full name of the OARK. At model level the principle is also the same, the nomenclature used is the names used of the relating OAs (in lower case) as previously explained separated by underscore, again with a possible prefix, infix and/or suffix. An example of an OAR is «A01-rental_starter.is_ an_initiator_of.T02-car_drop-off».

To help on the task of remembering all the names of the fact types, the Halo extension used. It is an extension to SMW and has been developed as a part of Project Halo in order to facilitate the use of Semantic Wikis for a large community of users. The focus of the development was to create tools that increase the ease of use of SMW features and advertise the immediate benefits of semantically enriched contents. We decided to use Halo due to its auto-completion feature that is a great help for the task of defining and reusing organization artifacts. This happens as, for example, actor roles and transactions names are frequently changed and Halo extension allows us to prevent inconsistencies in the specification and interpretation of the artifacts and their names. With Halo it's possible to define properties in a very clear manner to connect pages and create semantic relations between them.

**Template:DIAGRAM TEMPLATE**

This is the "DIAGRAM_TEMPLATE" template. It should be called in the following format:

```
{{DIAGRAM_TEMPLATE
|Diagram ID=
|Diagram Name=
|Is Instance of=
|Represented Model Kind=
|Diagram Description=
|Represented Shapes=
|Represented Connectors=
}}
```

Figure 9: DIAGRAM TEMPLATE page view.

## 3.4 Semantic Forms

Semantic MediaWiki offers us countless extensions, one of them being the Semantic Forms. Semantic Forms are of a substantial value in maintaining the correctness and the structure of the whole wiki pages. Semantic Forms allow for a full structural definition for all the pages of the same kind using three constructs; *properties*, *templates* and *forms*.

Properties are the elementary “construct” of semantic forms, and, for every piece of information in a SMW page, a property should be created. For example in the page «CA02-driver_shape» on SMW, we would have properties such as «actor_name» or «actor_id» as represented in Figure 8.

These properties are then grouped in Templates. Templates are in a basic way structuring the allowed properties for each page. In a concrete implementation of the UEAOM, there is the need for a template for each of the represented classes in the model in order for structured instances of those classes to be created. For example for the Object

Class Diagram, a template would list the «diagram_id», the «diagram_name», the «is_instance_of», the «represented_model_kind», the «diagram_description», the «represented_ shapes» and the «represented_connectors» as shown in Figure 9.

One would notice that in Figure 9, the names are not accordingly to our previously defined nomenclature nor the properties just mentioned. This is because the list in the Template page are not the properties themselves but instead, the label names we decided to give to each of them when creating the template. These labels are useful to maintain the pages user friendly but one could had simply used the same name. In the "edit page" option in the «DIAGRAM TEMPLATE» page we can find the corresponding properties to each label as shown in Figure 10. Although this Template is used to create instantiations of DIAGRAM, this is not the same thing as the class DIAGRAM KIND, such class still needs to exist as a page and with a template of its own. The template pages for the UEAOM classes can be seen as the SMW page implementation of the classes themselves.

The Forms are the implementations for the templates. For instance in a «DIAGRAM FORM» one would fill all the listed properties in the «DIAGRAM TEMPLATE», but this task is intended to also be allowed by creating a diagram using SVG that would automatically generate the template, leaving no need to use the forms for such creation. Forms however are still useful in an editing perspective, as one can edit the pages using the form instead of the visual editor, while keeping the data structured.

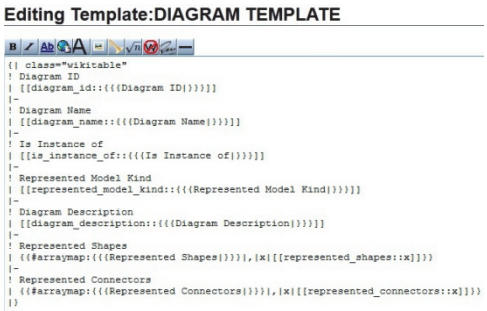



Figure 10: DIAGRAM TEMPLATE edit view.

# 4 CONCLUSIONS

With our SMW implementation presented in this paper we take advantage of semantic web technology to model and represent organizations. We were able to create wiki pages containing semantic properties, capable of storing in a formal and precise way, not only organizational knowledge in the form of structured organization artifacts, but also their representations, enabling also user friendly creation and change of models and diagrams by a Javascript GUI developed for this purpose. All diagrams are dynamically generated in run time, using stored semantic properties in the organization artifacts pages and in the SVG format, allowing easy visualization of all kinds of organization diagrams and also import/export to other tools. Using the SMW gives a huge versatility to our solution as semantic queries and sub queries allow us to detect and inform users on any violation of meta-model rules and to generate and endless pool of valid and interesting information regarding the organizational self. By using a wiki as a base, our solution inherently allows a community based effort of knowledge management regarding all kinds of aspects of organizations, not only using DEMO language but also any other modeling language, allowing also evolution and migration of existing models. Although already modeled conceptually in our UEAOM, the implementation of the specification of new versions of languages and model migration is currently still undergoing. As other future developments we expect to allow concurrent change in organization artifacts by multiple collaborators in a coherent and consistent way. We also expect to implement a semantic validation tool for runtime compliance checking with abstract syntax rules specified at meta-model level, informing the user of errors allowing an informed and user-friendly correction of models and diagrams.

# REFERENCES

Aveiro, D. & Pinto, D., 2013. Universal Enterprise Adaptive Object Model.

Bunge, M. A., 1979. *Treatise on basic philosophy, vol. 4, a world of systems*, Reidel Publishing Company.

Dietz, J. L. G., 2005. A World Ontology Specification Language. Em S. B. / Heidelberg, ed. *On the Move to Meaningful Internet Systems 2005: OTM Workshops*. pp 688–699. Available at: http://dx.doi.org/10.1007/11575863_88.

Dietz, J. L. G., 2006. Enterprise Ontology - Understanding The Essence Of Organizational Operation. Em C.-S. Chen et al., eds. *Enterprise Information Systems VII*. Springer Netherlands, pp 19–30. Available at: http://link.springer.com/chapter/10.1007/978-1-4020-5347-4_3 [Acedido Fevereiro 18, 2013].

Dietz, J. L. G., 2008. On the Nature of Business Rules. *Advances in Enterprise Engineering I*, pp.1–15.

Dietz, J. L. G. & Albani, A., 2005. Basic notions regarding business processes and supporting information systems. *Requirements Engineering*, 10(3), pp.175–183.

Halpin, T., 1998. Object-Role Modeling: an overview. Em *In http://www.orm.net/pdf/ORMwhitePaper.pdf*.

Hommes, B.-J., 2013. *ModelWorld*,

Magalhaes, R., Zacarias, M. & Tribolet, J., 2007. Making Sense of Enterprise Architectures as Tools of Organizational Self-Awareness (OSA). *Proceedings of the Second Workshop on Trends in Enterprise Architecture Research (TEAR 2007), June*, 6, pp.61–70.

Microsoft, 2010. *Visio 2010*, Microsoft.

MPRISE, 2010. *Xemod*,

Op' t Land, M., 2008. *Applying Architecture and Ontology to the Splitting and Allying of Enterprises*. TU Delft.

Saastamoinen, H. & White, G. M., 1995. On handling exceptions. *Proceedings of conference on Organizational computing systems*, pp.302–310.

Yoder, J. W., Balaguer, F. & Johnson, R., 2001. Architecture and design of adaptive object-models. *SIGPLAN Not.*, 36(12), pp.50–60.

Zacarias, M. et al., 2007. Towards Organizational Self-Awareness: An Initial Architecture and Ontology. Em P. Rittgen, ed. *Handbook of Ontologies for Business Interaction*. Information Science Reference, pp 101–121.