Universal Enterprise Adaptive Object Model

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, Adaptive Object Model, DEMO.

Abstract: In this paper we present a novel conceptual model that systematizes the integrated management and adaptation of: (1) enterprise models, (2) their representations, (3) their underlying meta-models, i.e., their abstract syntax and (4) the representation rules, i.e., concrete syntax for the respective models. All this for different modeling languages and also different versions of these languages. Thanks to our original use of the adaptive object model and type square patterns – normally applied in the context of software engineering, but here applied for enterprise engineering – we manage to provide a strong conceptual foundation for the development of software tools that will allow a precise and coherent specification of models and their evolution and also of meta-models and their evolution.

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 3 concerns: reality; models of reality; and their representations, while having adequate flexibility for model and meta-model evolution. In section 2 – Related Work & Problem – we present the basic notions for a proper understanding of our work as well as the problem itself and in section 3 – The Universal Enterprise Adaptive Object Model – we present our proposal contextualizing it and explaining in detail its principles and applicability. Finally, in section 4 – Conclusions – we briefly present our conclusions and discuss what contributions it brings.

Aveiro D. and Pinto D..
Universal Enterprise Adaptive Object Model.
DOI: 10.5220/0004550000890099
In *Proceedings of the International Conference on Knowledge Engineering and Ontology Development* (KEOD-2013), pages 89-99
ISBN: 978-989-8565-81-5

2 RELATED WORK & PROBLEM

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

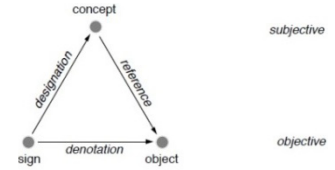

Figure 1: The meaning triangle.

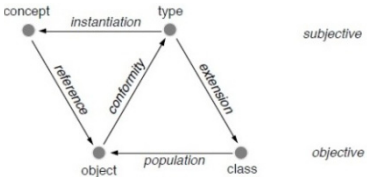

Figure 2: The ontological parallelogram.

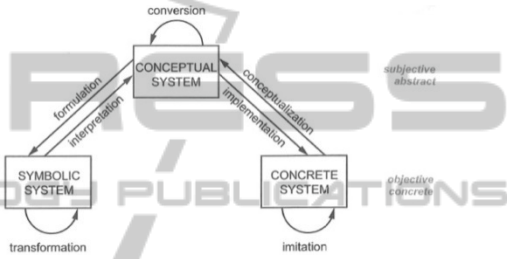

Figure 3: The model triangle.

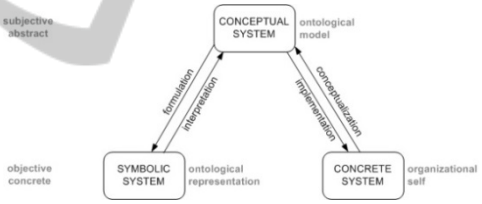

Figure 4: Model triangle applied to organizations.

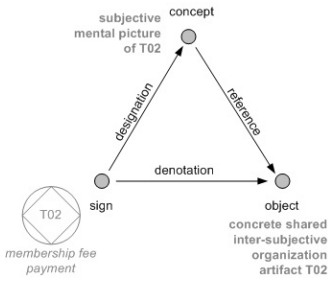

Figure 5: Meaning triangle applied to a transaction OA.

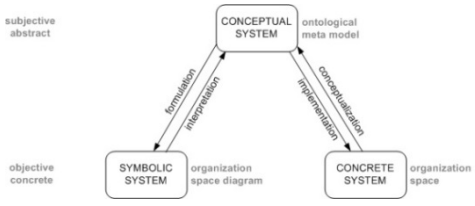

Figure 6: Model triangle applied to the organization space.

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, we find, 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.

From these notions we proposed a set of claims presented in more detail in (Aveiro et al., 2010) and summarized next. An organization – besides producing a set of products or services for its environment – also produces itself. That is, enclosed in its day-to-day operation, there will be parts of its operation which change the organization system itself, i.e., change the set of OAs that constitute its composition, structure and production. By formally and explicitly specifying these change acts one keeps a definite and updated record of produced OAs. Such a record – the OAs base – constitutes the means for one to always be able to conceptualize the most current and updated ontological model of the organizational self. Thus *the continuous production of the organizational self should include the synchronized production of the collective and subjective "picture" (awareness) of the organizational self – the conceptualization that constitutes its ontological model – thanks to the synchronized production of the respective symbolic system – an ontological representation that allows the interpretation of the ontological model and the conceptualization (awareness) of the organizational self*. To separate concerns, we propose that change acts are performed by a (sub-)organization considered to exist in every organization (O) that we call: *G.O.D. Organization* (GO) – change acts lead to the *G*eneration, *O*perationalization and *D*iscontinuation of OAs. The GO's production world will contain the current state of O's self as well as its relevant state change history. The GO has the role of continuously realizing and capturing changes of organizational reality. Thus, by implementing the GO pattern in a real organization, in an appropriate manner, providing automatic generation of ontological representations derived from the OAs base, one can achieve OSA. This is possible because one can implement clear rules that, based on the arrangement of OAs of the organizational self, automatically produce the appropriate ontological representation which, in turn, allows the appropriate interpretation of the ontological model, that is, the correct conceptualization of the organizational self.

OAs constituting the organizational self are arranged in a certain manner as to specify all the

spaces (state, process, action and structure) of an organization's world, i.e., they have to obey certain rules of arrangement between them. We call the specification of these rules as the *ontological meta model*. The ontological meta-model is the conceptualization of the *OA space*. By OA space we understand the set of allowed OAs. It is specified by the *OA base* and *OA laws*. The *OA base* is the set of *OA kinds* of which instances, called *OAs*, may occur in the state base of the GO's world. The *OA laws* determine the inclusion or exclusion of the coexistence of OAs. The definition of the OA space is quite similar to the definition of state space of an organization's production world – specified in World Ontology Specification Language (WOSL) (Dietz 2006) – and, thus, it is appropriate to use WOSL to express the ontological meta-model in, what we propose to call: the *Organization Space Diagram* (OSD). DEMO's OSD is currently called as the *DEMO Meta Model* (DMM), the chosen name for the specification provided in (Dietz, 2009) and consisting, in practice, in the OSDs corresponding to the four DEMO aspect models: SM, CM, PM and AM. These diagrams formulate, for each aspect model, the OA kinds out of which instances – OAs – can occur in the organizational self and coexistence rules governing how to arrange these instances. Another reason we propose to use the expression Organization Space Diagram is because we're in fact looking at a Space Diagram which, following the model triangle (Dietz, 2006), is a symbolic system which is a formulation of the conceptual system of the ontological meta model. So, for coherency reasons, one should not use terms “Meta” and “Model” to name those figures but use, instead, the term Organization Space Diagram. The OSD allows the interpretation, in one's mind, of the ontological meta model. The complete set of organization artifact kinds and laws governing the arrangement of their instances constitutes the organization space. The conceptualization of the organization space consists in the ontological meta-model which, in turn, is formulated in what we call the Organization Space Diagram. A depiction of this reasoning is present in Figure 6, another adaptation from the model triangle. The G.O.D. organization is addressed in detail in (Aveiro et al., 2010). The proposal presented in this current paper consists in an evolution of the conceptual model proposed in this other paper, taking in account state-of-the-art related model theory and concepts described next.

2.2 Theoretical Foundations on Models

In a graphical modeling language, the vocabulary is expressed in terms of pictorial signs. Those graphical primitives form the *concrete syntax* i.e the lexical layer of such language. The *abstract syntax*, on the other hand, is usually defined in terms of an abstract visual graph or a meta-model specification. A meta-model specification of a language defines the set of grammatically correct models that can be constructed using that language, a vocabulary. The concrete syntax provides a concrete representational system for expressing the elements of that meta-model (Guizzardi, 2005). In a communication process, besides agreeing on a common vocabulary, the participants need to also share the meaning for the syntactical constructs being communicated so they are able to interpret in a compatible manner the expressions being used. To this end a language's semantics can be constructed in two parts: a semantic domain i.e. the real world entities to which those semantics apply and a semantic mapping from the syntactic vocabulary to such domain that tells us the meaning of each of the language's expressions as an element in that specific domain. In graphical languages, vocabulary, syntax and semantics cannot be clearly separable. A graphical vocabulary of a modeling language may include shapes of differing sizes and colors that often fall into a hierarchical typing that constrains the syntax and informs about the semantics of the system (Guizzardi, 2005). The abstract syntax of a model manages the formal structure of the model elements and the relationships amongst them (La Rosa et al., 2011).

The MetaObject Facility (MOF) Specification is the industry-standard environment where models can be exported from one application, imported into another, transported across a network, stored in a repository and then retrieved, rendered into different formats like XMI or XML, transformed, and used to generate application code (OMG, 2012). The Adaptive Object Model (AOM) is a pattern that represents classes, attributes, and relationships as meta-data. It is a model based on instances rather than classes. Users change the meta-data (object model) to reflect changes in the domain. These changes modify the system’s behavior. In other words, it stores its Object-Model in a database and interprets it. Consequently, the object model is active, when you change it, the system changes immediately (Yoder et al., 2001).

3 THE UNIVERSAL ENTERPRISE ADAPTIVE OBJECT MODEL

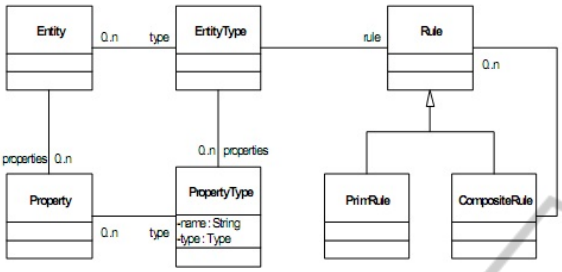

Figure 7: Type Square.

The long term objective of our research is the development of a wiki-based system that allows an effective integrated enterprise modeling, while allowing dynamic evolution of meta-models, models and their representations, while providing intuitive navigation through their elements and also their semantics, allowing wide-spread model interpretation and distributed model creation and change, reflecting enterprise changes, thus addressing our problem. An essential step in this direction is what we call the Universal Enterprise Adaptive Object Model (UEAOM), depicted in Figure 8. We apply the AOM pattern referred in the previous section so that each page or semantic property of our semantic wiki-based system corresponds to instances of classes of our AOM.

Wiki pages, that are instances of class DIAGRAM, automatically generate SVG diagrams based on shape and connector pages. These pages also allow dynamic editing of diagrams and underlying models. We also apply the type-square pattern (Yoder et al., 2001) – depicted in Figure 7 – 4 times as to allow run-time dynamic change of: (1) meta-model elements, (2) model elements, (3) shape elements and (4) connector elements. Our UEAOM is represented with the World Ontology Specification Language (WOSL) (Dietz, 2005). WOSL is based in Object Role Modeling language (Halpin, 1998) which is also used as a base for the specification of the anatomy of Archimate (Lankhorst et al., 2010), a similar effort to ours. In (Ferreira et al., 2008) a relation between Adaptive Object Model pattern and the MOF standard is presented, where run-time instances of the operational level are equivalent to MOF's M0 and knowledge level; classes, attributes, relations and behavior is equivalent to M1, being M2 an equivalent to the models used to define an AOM. As in the work of Ferreira et al., in our UEAOM all these MOF levels are projected as run-time instances. In our prototype system, we have as instances both organization artifacts – i.e., concrete organization models – and organization artifact kinds – i.e., the meta-model specification or, in other words, the abstract syntax. So both M1 and M2 levels of the MOF framework exist and change at run-time. But the MOF and Ferreira's initiative are too software development oriented and too complex for our needs. Our main contribution in this paper is to apply these fundamental theoretical foundations and adapt them to the field of enterprise ontology.

Having the UEAOM contextualized, an explanation of its content is now due. With the UEAOM's classes we are not explicitly specifying syntaxes of particular modeling languages. What we can do, while instantiating these classes, is to specify any syntax of any modeling language, along with particular models of each language, and also their evolution, all this in run-time. For a better understanding and following the essential and important validation by instantiation principle (Dietz, 2009) we present, for all elements of our AOM, example instances for the DEMO language, namely a fragment of the EU-rent case's Construction Model and its respective Actor Transaction Diagram. Thus, we can find, in red color expressions, instances of both our classes and fact types of our UEAOM concerning the EU-rent case which allow a better interpretation of our proposal.

3.1 Abstract Syntax

Relevant classes for the specification of the abstract syntax of any version of any language are presented in Figure 9. The main concepts of the abstract syntax specification are expressed in the classes LANGUAGE, MODEL KIND, ORGANIZATIONAL ARTIFACT KIND (OAK) and ORGANIZATIONAL ARTIFACT RELATION KIND (OARK). They specify all allowed artifacts (e.g. transaction kind OAK and transaction execution relation OARK) for different types of models that can exist for different languages. Class ORGANIZATIONAL ARTIFACT RELATION KIND has ten properties that can be divided in two groups of five where each group specifies one of the two sides of an allowed relation between two OAKs. The ones named prefix, infix and suffix specify the formulation that can be done around the names of the two OAKs being related. Most times, only the infix needs to be specified. With the unicity and dependency properties we specify the cardinality of

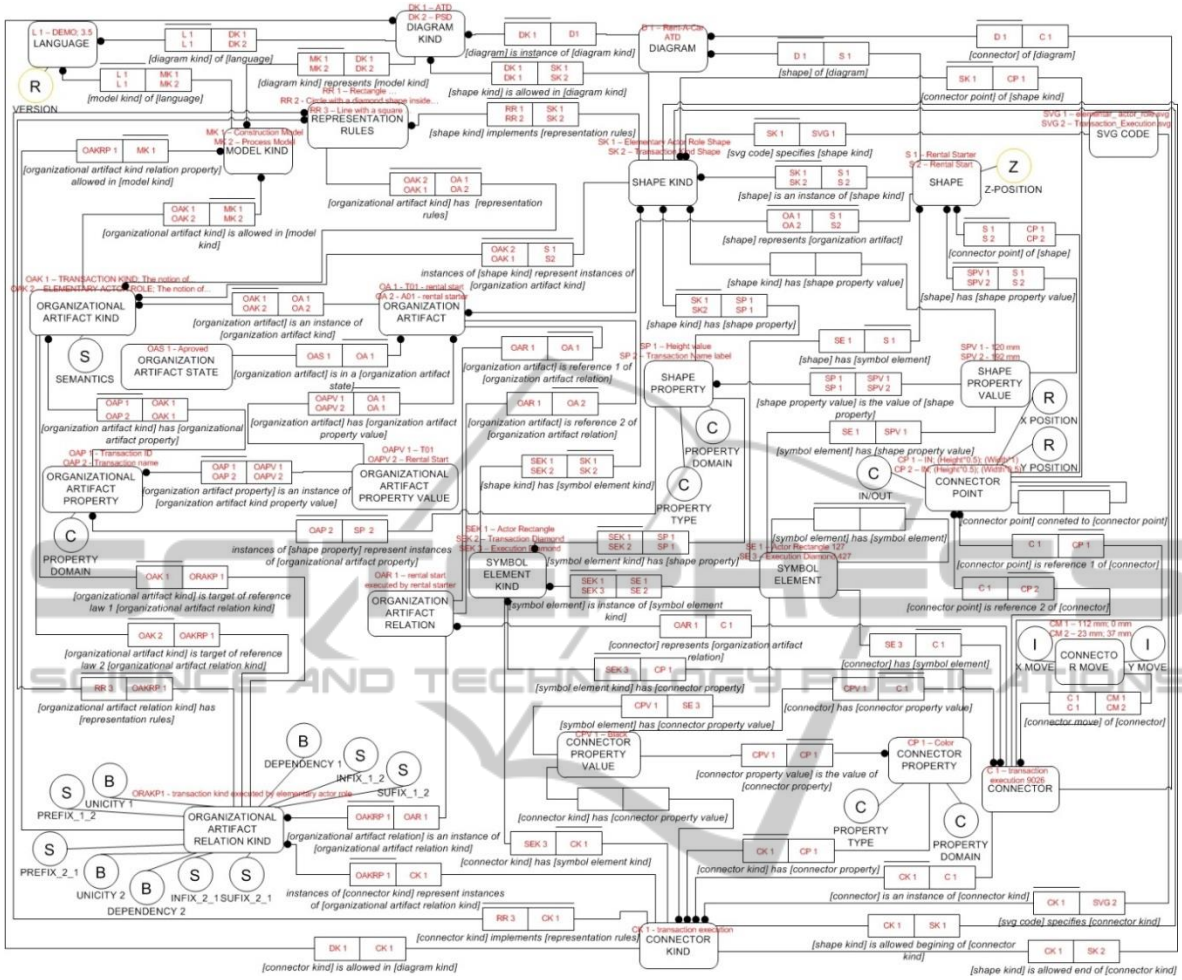

Figure 8: Universal Enterprise Adaptive Object Model.

the relation and which OAKs are mandatory or not to participate in the relation. *Reference law* fact types specify which two OAKs are allowed to participate in this relation. Practical example of the first set of the referred 5 properties: F *Transaction Kind* T ***is initiated by*** *Elementary Actor Role* corresponds to a set of Dependency 1, Reference law 1, Unicitiy 1, Infix_1_2 and Reference law 2. F *Elementary Actor Role* T ***is initiator of*** *Transaction Kind* would be its corresponding Dependency 2, Reference law 2, Unicitiy 2, Infix_2_1 and Reference law 1. Thanks to this part of our UEAOM specification we allow a precise and formal formulation of the abstract syntax of models, already giving considerable semantics thanks to the prefix, infix, suffix and OAK names that can be composed in formulations for each direction of the relation. Instances of class ORGANIZATION ARTIFACT PROPERTY specify intrinsic properties of OAKs, like identifiers and names. The respective property PROPERTY DOMAIN allows us to specify the domain for each intrinsic property of an OAK (e.g., string, number, etc.). Examples of instances are property *transaction id* with domain *T<number>* or *transaction name* with domain *<string>*.

In Figure 10 we can see an excerpt of the current DEMO ontological meta-model and the UEAOM classes used to define it. Both these models are the equivalent to the M2 MOF model that, as we have seen, sets the rules for specifying concrete models. All elements of this meta-model can be considered instances of the classes we just have presented. The binary fact type *[elementary actor role] is an initiator of [transaction kind]* is, in our UEAOM, an instance of ORGANIZATIONAL ARTIFACT RELATION KIND class, with values for the infixes being: *initiates* and *initiated by*. There are, however, other classes: DIAGRAM KIND, SHAPE KIND, CONNECTOR KIND, CONNECTOR and SHAPE PROPERTY that are present in this Figure 10 and

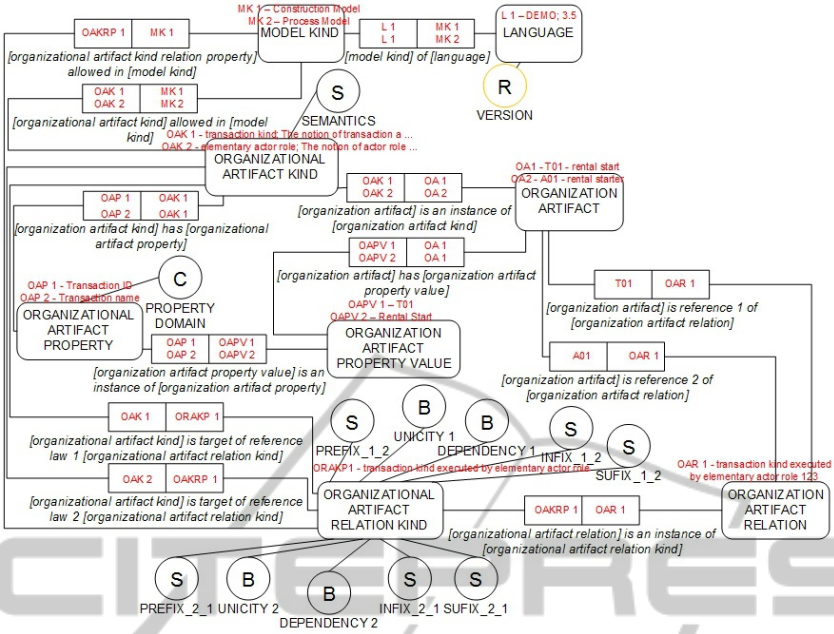

Figure 9: UEAOM - Abstract Syntax classes.

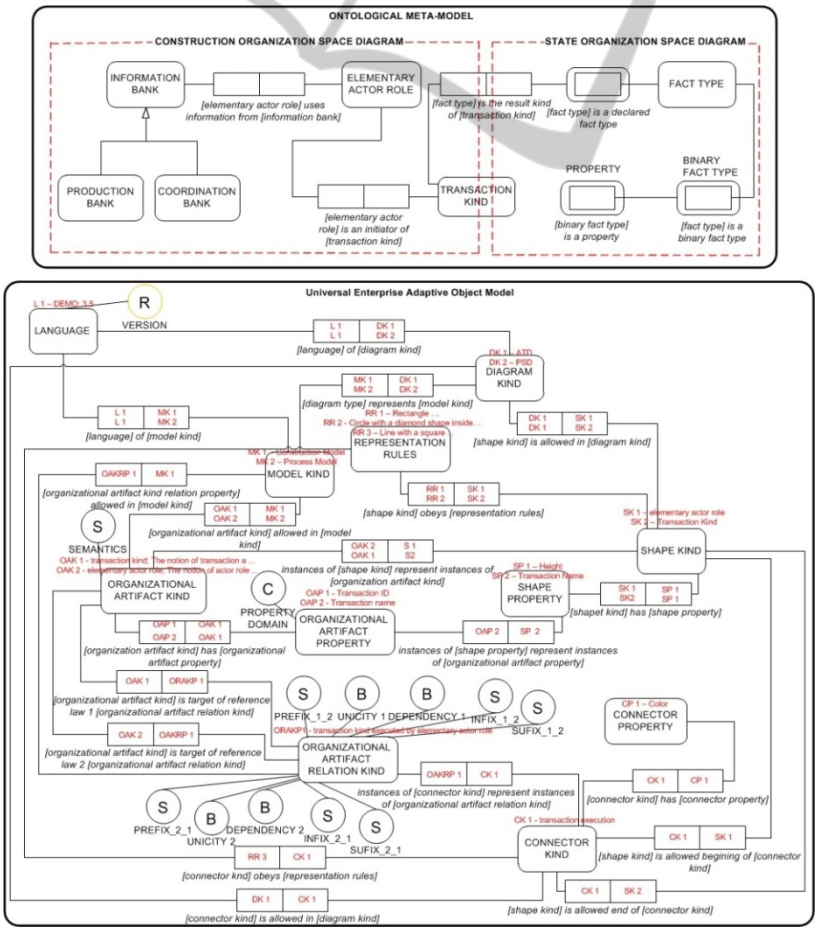

Figure 10: DEMO Ontological Meta-Model and UAOM classes used to represent it.

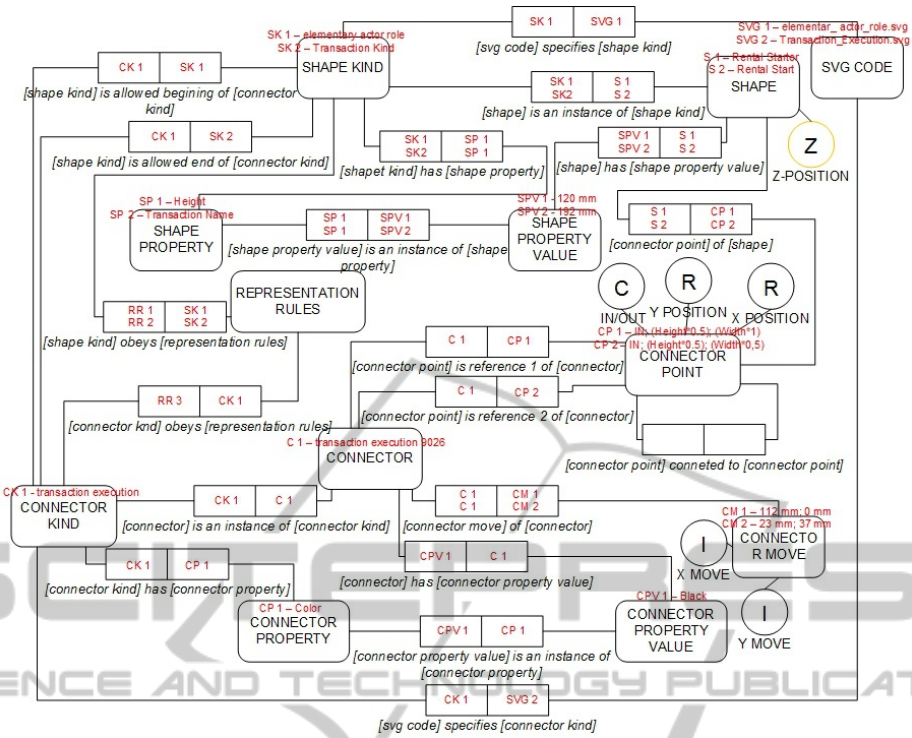

Figure 11: UEAOM - Concrete syntax.

are part of the meta-model level of the UEAOM but are not part of the abstract syntax, these will be explained in more detail in the next section.

3.2 Concrete Syntax

The UEAOM classes that allow the specification of rules for the concrete representation of models, i.e., the concrete syntax, are presented next. These classes, together with all their inter-relating fact types are present in Figure 11. With the class SHAPE KIND, instances of the types of shapes allowed to be part of diagram kinds representing certain model kinds are specified. These shape kinds are also specifically connected to the OAKs whose instances they will represent. For example, the *elementary actor role shape* is allowed in diagram kind *Actor Transaction Diagram*, that represents the *construction model* of DEMO language. Instances of this shape represent instances of OAK *actor role*.

With SHAPE PROPERTY, we specify the properties for each shape, e.g., *line color* and *actor id label* of actor role shape. Instances of CONNECTOR KIND specify allowed representations for OAKRs, e.g. *transaction initiation connector* instances represent instances of OAKR *transaction initiation*. With CONNECTOR PROPERTY, the properties of each connector are specified, e.g., for the just mentioned connector, *line color: black* and *line dashing: continuous*.

Instances of REPRESENTATION RULES, class are an informal textual based specification of rules on how ORGANIZATIONAL ARTIFACT KINDS and ORGANIZATIONAL ARTIFACT RELATION KINDS should be represented. These rules are taken in consideration in either SHAPES or CONNECTORS that represent those OAKs and OARKs. For example, a transaction is a black circle with a black diamond inside. It is also according to the REPRESENTATION RULES that we have a definite answer if an OARK will give origin or not to a connector or if instead it will be represented by the connection of two shape kinds directly. Revisiting the full example from Figure 8, an *elementary actor role shape* would be an instance of class SHAPE KIND, for the representation of instances of the *actor role OAK*. *Transaction shape* would also be an instance of SHAPE KIND for the representation of instances of *transaction OAK*. So an instance of class CONNECTOR KIND for the representation of this OAKR would be *transaction initiator connector*, with properties like *line type: dashed*. Many of the SHAPE KINDs and CONNECTOR KINDs are comprised by multiple symbols that need to be considered individually as having a set of properties. Although in most cases

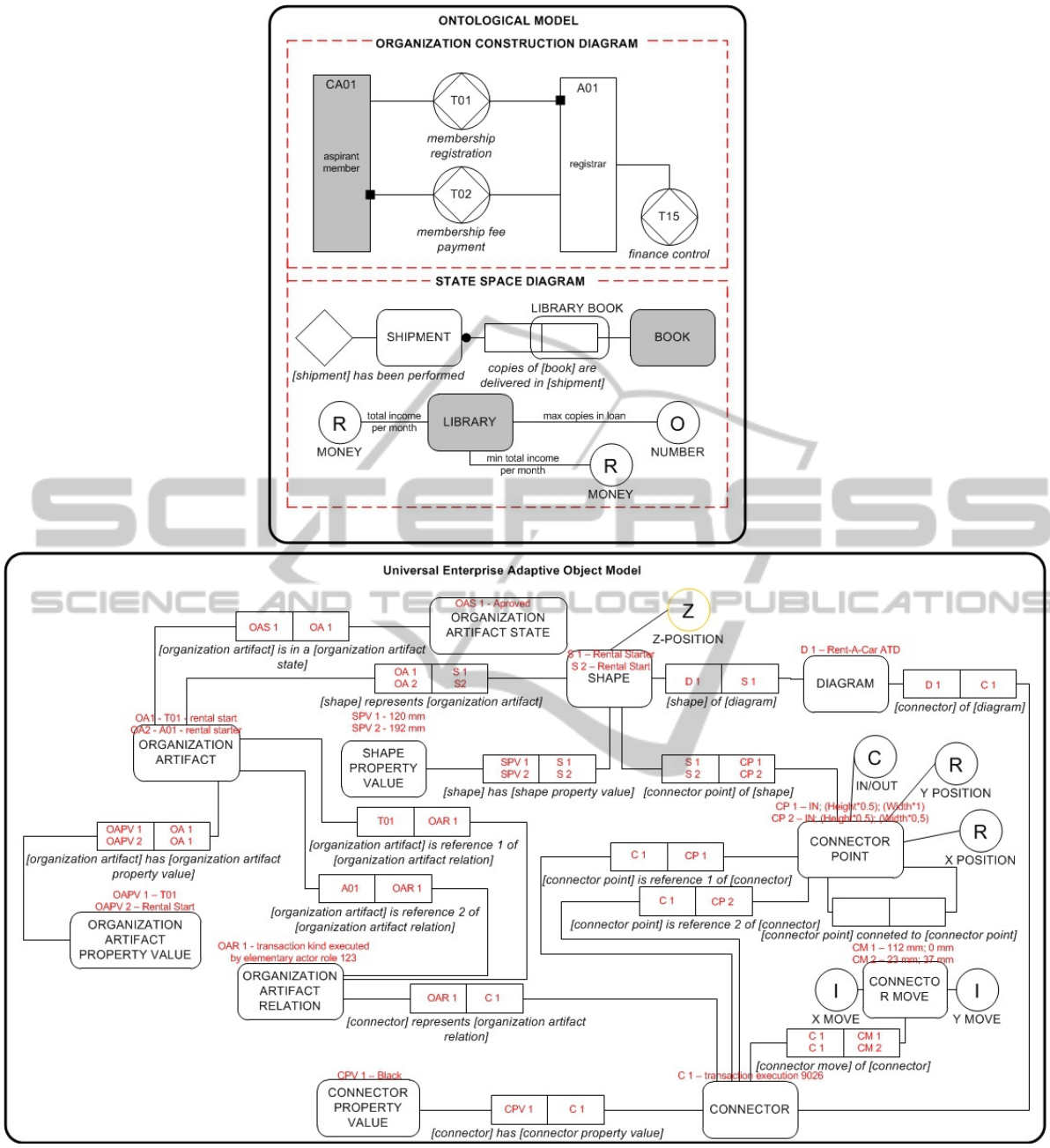

Figure 12: DEMO concrete diagrams example and UEAOM classes used to represent them.

the aggregate of composing symbols are treated as "one" in the diagram drafting, such as a circle and diamond in an actor transaction diagram transaction, that have a fixed size (height and width) and none of them can be altered, there are also cases in which symbols need to be treated and moved in the diagrams in a separate and independent way having their own set of SHAPE PROPERTIES or CONNECTOR PROPERTIES like, for example, in a process step diagram where the diamond inside the transaction can be moved and re-sized according to the needs. As a solution for this, we have classes SYMBOL ELEMENT KIND that specify each symbol element to be present in a shape kind or connector kind and SYMBOL ELEMENT that are instances of SYMBOL ELEMENT KIND and specify concrete representations of SYMBOL ELEMENTS of a specific kind. As an example of this we can consider the actor transaction diagram SHAPE KIND transaction as being composed by the

SYMBOL ELEMENT KINDS *Transaction Diamond* and *Transaction Circle*.

In Figure 12 we have a partial example of a concrete representation of the DEMO ontological models of an Actor Transaction Diagram and Object Fact Diagram and the corresponding part in the UEAOM. DEMO Ontological models are the equivalent to the M1 level of MOF and instances of their OA's to the MOF's M0 level.

Ontological Models and their representation are covered in the UEAOM by the classes: DIAGRAM, where concrete instances of a certain DIAGRAM KIND are specified; SHAPE, where concrete instances of SHAPE KIND are specified; SHAPE PROPERTY VALUE, where concrete instances of SHAPE PROPERTY are specified; CONNECTOR, where concrete instances of CONNECTOR KIND are specified; CONNECTOR PROPERTY VALUE, where concrete properties of the CONNECTOR PROPERTY are specified; ORGANIZATIONAL ARTIFACT, where concrete instances of ORGANIZATIONAL ARTIFACT KIND are specified; ORGANIZATIONAL ARTIFACT PROPERTY VALUE, where concrete OA properties are specified and ORGANIZATIONAL ARTIFACT RELATION, where concrete instances of ORGANIZATIONAL ARTIFACT RELATION KIND are specified and all their relating fact types. In this way also allowing them to be changed in an easy and consistent way in run-time environment.

Again using a concrete example from Figure 12, we have the “CA-01 aspirant member shape”, this is an instance of SHAPE (this SHAPE an instance itself of the SHAPE KIND “Composite Actor Role”) that represents the ORGANIZATIONAL ARTIFACT “CA-01 aspirant member” (itself an instance of the ORGANIZATIONAL ARTIFACT KIND “Composite Actor Role”); the string “aspirant member” is an instance of SHAPE PROPERTY VALUE (that represents the instance of ORGANIZATIONAL ARTIFACT PROPERTY VALUE “aspirant member”) and so is “CA-01”. These two strings are VALUES, instances of the SHAPE PROPERTIES “Actor Name” and “Actor ID” respectively (that again represent the instances of ORGANIZATIONAL ARTIFACT PROPERTY VALUE “Composite Actor Name” and “Composite Actor ID”).

The DIAGRAM KIND and MODEL KIND classes were not present in the original DEMO Ontological meta-model as all models were specified in this single meta-model. But in our UEAOM, as we have generalized this definition to accommodate any language for organizational modeling, the DIAGRAM KIND and MODEL KIND classes are vital so we can relate to each specific Ontological Model. Actor Transaction Diagram or Process Step Diagram would be examples of instances of this DIAGRAM KIND while the first would be a representation of the MODEL KIND Construction Model and the second a representation of the MODEL KIND Process Model. The LANGUAGE class and its property VERSION is used to define the modeling language being modeled and the version of such language.

4 CONCLUSIONS

In this paper we proposed what we call the *Universal Enterprise Adaptive Object Model*, (UEAOM) that, through the multiple application of the type square pattern and the adaptive object model pattern, allows a robust and precise conceptual solution to manage changes in organization artifacts, their models and meta-models, all in run time environment. It is possible, through specification of instances of our UEAOM, to manage the various aspects of modeling languages: the semantics; the abstract and concrete syntaxes and the pragmatics all in the same implementation and object model. This gives flexibility for a very important contribution of our proposal: the possibility to completely change or create a new behavior for the system by changing its concrete and/or abstract syntax on the fly.

As we can see, important and essential representation aspects like shapes, connectors and their properties and representation rules, are not specified formally and precisely (as we do in our examples) in DEMO's original meta-model, nor in the meta-model of other mainstream languages such as BPMN or Archimate. The abstract and concrete syntaxes as well as the language's pragmatics are very important aspects of the modeling activity and should be specified in the meta-model specification itself, even if with semi-formal textual information (in our case provided by instances of the class REPRESETATION RULES) as to allow the least ambiguity possible. Our UEAOM approach to enterprise modeling differentiates itself from other enterprise modeling languages like BPMN and Archimate in the way that in expands the AOM principles to also model the meta-model level in run time environment taking as an advantage also of the type square pattern in a coherent, consistent and robust way. Our conceptual solution offers tool makers a sound theoretical base for an extensive and

thorough management of knowledge of an organization and also of the languages being used to create models. It also considers the DEMO's Ψ-theory principle that nothing ceases to exist, but instead, artifacts have their state changed. Although specified with the implementation of the DEMO methodology in mind, the UEAOM is flexible enough to allow the modeling of multiple languages like Archimate or BPMN.

As future developments of this work, we will provide a more detailed specification of how to manage the versions of the language and the possibility to migrate models and also a better specification of the state changes that can occur with all artifacts that are instances of our UEAOM classes.

REFERENCES

Aveiro, D., Rito Silva, A. & M. Tribolet, J., 2010. Extending the Design and Engineering Methodology for Organizations with the Generation Operationalization and Discontinuation Organization. Em *5th International Conference, DESRIST 2010*. St. Gallen, Switzerland, June 4-5, 2010: Springer, pp 226–241.

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., 2009. Demo meta model specification (forthcoming, in www.demo.nl).

Dietz, J.L.G., Enterprise ontology: theory and methodology. Springer-Verlag New York, Inc. Secaucus, NJ, USA. (2006).

Dietz, J. L. G., 2009. Is it PHI TAO PSI or Bullshit? Em The enterprise engineering series. Methodologies for Enterprise Engineering symposium. Delft: TU Delft, Faculteit Elektrotechniek, Wiskunde en Informatica.

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.

Ferreira, H. S., Correia, F. F. & Welicki, L., 2008. Patterns for data and metadata evolution in adaptive object-models. Em *Proceedings of the 15th Conference on Pattern Languages of Programs*. PLoP '08. New York, NY, USA: ACM, pp 5:1–5:9. Available at: http:// doi.acm.org/ 10.1145/ 1753196.1753203.

Guizzardi, G., 2005. Ontological foundations for structural conceptual models. Available at: http://doc.utwente.nl/ 50826/.

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

Lankhorst, M. M., Proper, H. A. & Jonkers, H., 2010. The Anatomy of the ArchiMate Language. *International Journal of Information System Modeling and Design*, 1(1), pp.1–32.

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.

OMG, 2012. OMG's MetaObject Facility (MOF) Home Page. Available at: http://www.omg.org/mof/.

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

La Rosa, M. et al., 2011. Managing Process Model Complexity Via Abstract Syntax Modifications. *IEEE Transactions on Industrial Informatics*, 7(4), pp.614–629.

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.