Aplicação de Ontologias à Representação e Análise de Modelos de Arquitetura Empresarial

André Coutinho Morais

andre.coutinho@ist.utl.pt

Instituto Superior Técnico, Universidade Técnica de Lisboa, Portugal

Abstract: Enterprise architecture supports the analysis, design and engineering of business-oriented systems through multiple views. Each view expresses the elements and relationships of a system from the perspective of specific system concerns relevant to one or more of its stakeholders. As a result, each view needs to express in the architecture description language that best suits its concerns. Therefore, enterprise architecture may be described using a set of different languages. Also, the existence of semantic gaps between enterprise architects and stakeholders may produce conceptual misalignments. However, current enterprise architecture modeling languages display two issues in this setting. First, they lack mechanisms to integrate multiple architecture description languages. This issue hinders the specification of views using different languages. Second, enterprise architecture models lack computable analysis support. A new set of solutions will be proposed by applying an ontology-based approach to specify and integrate multiple architecture modeling languages and to analyse the resulting integrated models.

Keywords: enterprise architecture, ontology, conceptual modeling, model analysis, ArchiMate, OWL

1. Introduction

Enterprise architecture (EA) is defined by Lankhorst as "a coherent whole of principles, methods, and models that are used in the design and realization of an enterprise's organizational structure, business processes, information systems, and infrastructure", providing a basis for business-IT alignment (Lankhorst, 2006). Such alignment is typically achieved through the creation of models and other artifacts describing different aspects of the organisation, including business and IT elements. Managing the dependencies between and within the different models and other artifacts produced during the application of architecture development methods is crucial for supporting the communication between the different stakeholders, and the alignment and consistency between the development artifacts and other products (Galster, 2011) (Romero, Jaen, & Vallecillo, 2009). In addition, the management of the enterprise architecture requires the ability to analyze the articles so as to assist business analysis (Galster, 2011)

1.1. Related Work

Several authors reported that comprehensive approaches or horizontal building architectures fail to capture aspects that pertain to specific areas of the organization (Saat, Franke, Lagerström, & Ekstedt, 2010) (Buckl, Matthes, & Schweda, 2010). Part of the reason for this stems from the fact that a system has multiple stakeholders and these have different concerns about the system, and therefore specific requirements. Therefore need different views of the architecture to respond to the various stakeholders. This is exactly the approach defined in ISO 42010 (IEEE, 2011) that designates multiple views are needed to describe an architecture. An Architecture Descriptor requires multiple views, generating a set of models that represent viewpoints related to the interests of stakeholders. However, the creation of these viewpoints raises a new challenge as it becomes necessary to ensure the traceability of concepts between different views as well as ensuring their consistency, i.e, the uniform use of the same concept in different views.
Analysis performed by experts are more flexible approaches, but also most time-consuming, and depend on the experience of the person who is doing the analysis. Additionally these people base their analysis on views generated and the result are generally counties and ideas for future developments. Analysis based on rules can be automated and able to validate compliance with model rules (Buckl, Matthes, & Schewda, Classifying Enterprise Architecture Analysis, 2009).

1.2. Problems

To proceed with the analysis of models of an enterprise architecture, i.e, able to respond to concerns relating to a range of stakeholders, there must be traceability and consistency between models generated during the construction of the architecture. Somehow the ArchiMate try to remedy these problems and improve this traceability between different domains (M. Lankhorst, 2004), but cannot represent the whole architecture with the degree of detail required by a stakeholder due to some specific organizational domains face the generality of domain-independent models (Saat, Franke, Lagerström, & Ekstedt, 2010).

Describe each domain in a specialized language and then integrate these multiple languages causes problems. The lack of integration mechanisms between these languages hinders the traceability and consistency checking in models built. Finally exist the difficulty of maintain a semantic relationship between elements of the models.
In this sense it is necessary to find mechanisms to integrate multiple EA languages and thus find the answers to the following problems.

1. **What techniques can be used to specify models for enterprise architecture?**
2. **What techniques can be used to integrate multiple domains of enterprise architecture?**
3. **How to make analysis of models of enterprise architecture?**

2. Problem Analysis

Generally, the problem is defined as finding a way to represent EA models in a formal way, i.e, to provide computational analysis in models, enabling with this formality, advantages in the construction, integration and analysis of these models. More specifically it is focused on representing a language that has a highly graphic characteristic, ArchiMate, in a structural feature language, the OWL2.

It is important to realize that there are different kinds of information display, for example, text, tables, graphics models, among others. Each representation has its reason of being and ease of interpretation depending on the information to be displayed and the type of analysis that will be done about this information.

The EA shall support the governance and decision support (Lankhorst, 2006), so it should get as much useful information as possible from the knowledge contained in the models, which can reach a great level of complexity if is elaborated in great detail (Binz, Leymann, Nowak, & Schumm, 2012). Such analysis can be performed without computational support but becomes more difficult as the analysis becomes more profound and in complex scenarios.

Different organizations have different business needs and operate in different markets, and have different demands for information that seek to obtain in EA models. This makes it challenging to create a model that can meet all these needs.

With the ArchiMate is possible to create several models, these models represent specific interests of each stakeholder. The number of models built depends on the needs of stakeholders.

The EA models can be constructed using different languages with different notations.

In integration of models context some of the problems that can arise are:

- Coherence of models
- Consistency between the models
- Traceability between models

These problems become even more critical when these models evolve over time.

The models created with ArchiMate, due to their visual appearance, inhibit automatic processing, automatic treated here in a functional sense, which providing an entry (i.e. a question) to model, it is possible to automatically obtain an output (i.e. a model

element). Thus the analysis of one or more models of the ArchiMate can become costly as they involve the analysis of information that is not directly explicitly. For models created with different languages there must be a logical connection, which can be processed computationally, relating these models.

Still on the issue of processing, we may have some difficulty in validations of the models, for example to validate if the models are created according to the Archimate standard. In this sense, transforming the models into something that is possible to perform some processing on these, we have a manner to perform these validations in an automatic or semiautomatic way.

The models of the ArchiMates do not have a representation scheme that can be processed, helping analysis in their models, it is essential to process the information instead of just presenting.

The core of ArchiMate language, which is its meta-model, has the basic concepts and relations that serve for modeling general purpose EA, however the language should be able to facilitate, through extension mechanisms and specializations, or domain-specific purposes such as: (Group, 2012)

- Support for specific types of analysis models.
- Support for communication between architectures
- Capture the specifics of a applicational domain

The argument behind this statement is to provide a means to allow extensions of the core language that are tailored towards such specific domains or applications, without burdening the core with a lot of additional concepts and notation which most people would barely use. Thus, by working with the ontology, we need to be able to maintain this property, so that we can make the extension of ArchiMate ontology as well be able to relate this to others ontologies without overloading the core language.

3. Proposal

In order to ground our proposal in good practice and allow for a structured and extensible approach, a set of architecture principles were defined. As such, the following design principles were considered.

- Concern orientation: The architecture shall represent the concepts necessary and sufficient to address an explicit set of modeling concerns. This means that the model shall be derived from the questions that need to be addressed and to provide answers to those questions. This also means that the model shall not support any concepts that are not explicitly derived from stakeholder concerns.
- Expressiveness: The architecture shall be able to represent the domain concepts without ambiguity. This entails defining the minimum set of types and relationships to describe a domain.
- Extensibility: The architecture must cope with extensions because context modeling entails using multiple concurrent perspectives on the same problem. This derives from being able to answer to multiple concerns. Therefore, domain-specific and domain-independent models must coexist and the overall architecture must cope with multiple model transformation and integration. A

specific concern is that the architecture is extensible to new application domains.

- Viewpoint-orientation: The architecture must support defining views over subsets of its concepts. This serves to facilitate the communication and the management of the models as viewpoints act as a separation of concerns mechanism. Viewpoints will facilitate addressing multiple concerns and can improve decision-making by isolating certain aspects of the architecture according to the needs of decision makers.
- Modularity: The architecture must follow the principles of high-cohesion and low-coupling. Observing these principles contributes to expressiveness and extensibility of the architecture. It is especially important that adding new domain-specific aspects to the model does not interfere with the concepts already present in the model.

Following these principles, our approach is based on the use of a core meta-model, also termed domain-independent ontology (DIO), which represents a domain-independent language (i.e., that does not address any specific domain-dependent concerns), containing the minimum set of concepts required for addressing the majority of scenarios. Then, that simple core meta-model can be extended in a plug-in fashion with other domain-specific meta-models, in a varying number depending on the situation at hand, which are termed domain-specific ontologies (DSOs). Each DSO represents a domain-specific language that addresses a particular set of concerns, and should also have the minimum set of concepts required for describing a determined domain. Low-coupling and high-cohesion are the principles guiding the addition of new DSOs: the number of dependencies between the DSOs and DIO should be minimal, and each DSO should deal only with a set of domain-specific concerns. In this way, the addition of DSOs should have a minimum impact on the DIO and existing DSOs. For achieving traceability between the DIO and the DSO, it is necessary to integrate the ontologies. Thus, ontology integration deals with the combination of the different ontologies for ensuring consistency and maximum coverage of the domain being addressed. The simplest case is that of integrating the DSO with the core concepts represented in the DIO. Cross-DSO integration can also occur in cases where there is the need for adding more expressive power to specific domains. The ontology integration makes use of model transformation, which involves defining a mapping strategy from a source model to a target model.

Considering the mapping between a DIO and a DSO, it might be as simple as a 1:1 correspondence between the concepts of the two ontologies. However, it is expected that mapping deficiencies might occur, as domain-specific languages might contain very specific concepts that are not mappable at all into the DIO. Using the Bunge-Wand-Weber representation model as an inspiration [28], one concept on the DSO might map to several concepts in the DIO (overload), a concept in the DSO might not map at all to the DIO (deficit), or, in the least common case, several concepts in the DSO might map to one concept in the DIO (redundancy). In this paper we are not dealing with this challenging aspect of ontology integration. Nonetheless, it is assumed that the mapping between some of the concepts of the DSO and some of the concepts of the DIO will always be possible as the kinds of problems we are dealing are situated in the information systems domain.

Redundancy, overload and deficit cases are thus expected to occur frequently. Cases where doubts might occur concerning the mapping between two concepts are carefully judged, with the mapping not being considered in cases of incompatibility.

The architecture makes possible the usage of reasoning for performing analysis of the models: the core DIO and the extension DSOs. Four analysis configurations are possible:

- Intra-DIO reasoning, when inference is limited to the concepts of the DIO;
- Intra-DSO reasoning, when inference is limited to the concepts of the DSO;
- cross-DSO reasoning, whenever a mapping transformation between different DSO is available, inference can use concepts from different DSO;
- cross-DIO-DSO reasoning, when inference uses concepts from both the DIO and one or more DSO, requiring a mapping transformation between each DIO-DSO pair.

Figure 1 shows the architecture of the solution.

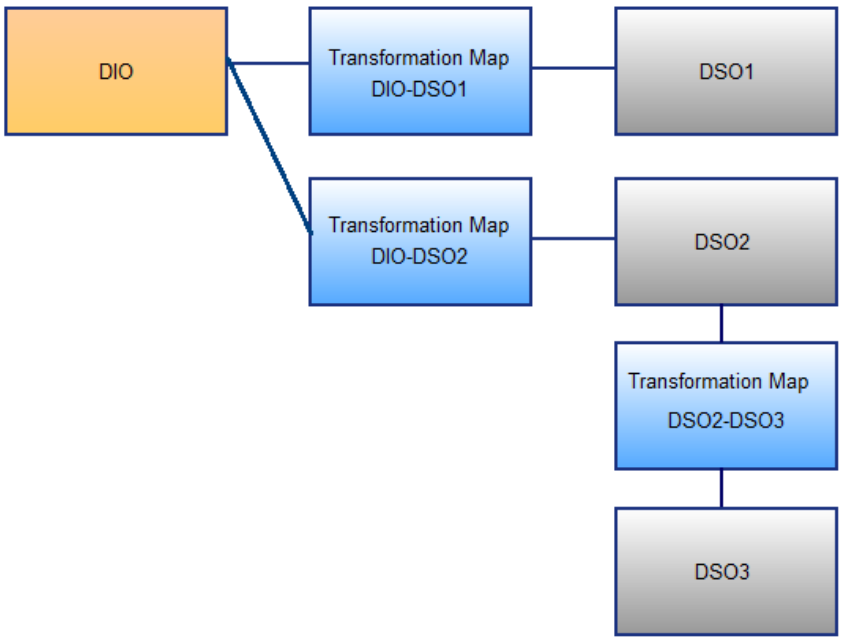

Figure 1 – Architecture of the solution

4. Case Study

4.1. ArchiMate ontology

An analysis of ArchiMate's meta-model was performed concept-by-concept, including the relations with other concepts and the constraints existing in those relations. The result was the mapping of concepts into OWL classes and the mapping of relations into OWL ObjectProperties. Restrictions were added to the properties, such as InverseObjectProperties and SuperObjectProperties axioms were added to the OWL ontology, so that derived relationships can be inferred.

Cardinality was also added to ensure the compliance against the ArchiMate specification shows some of the aspects of ArchiMate. Figure 2 depicts a subset of the ontology restrictions.

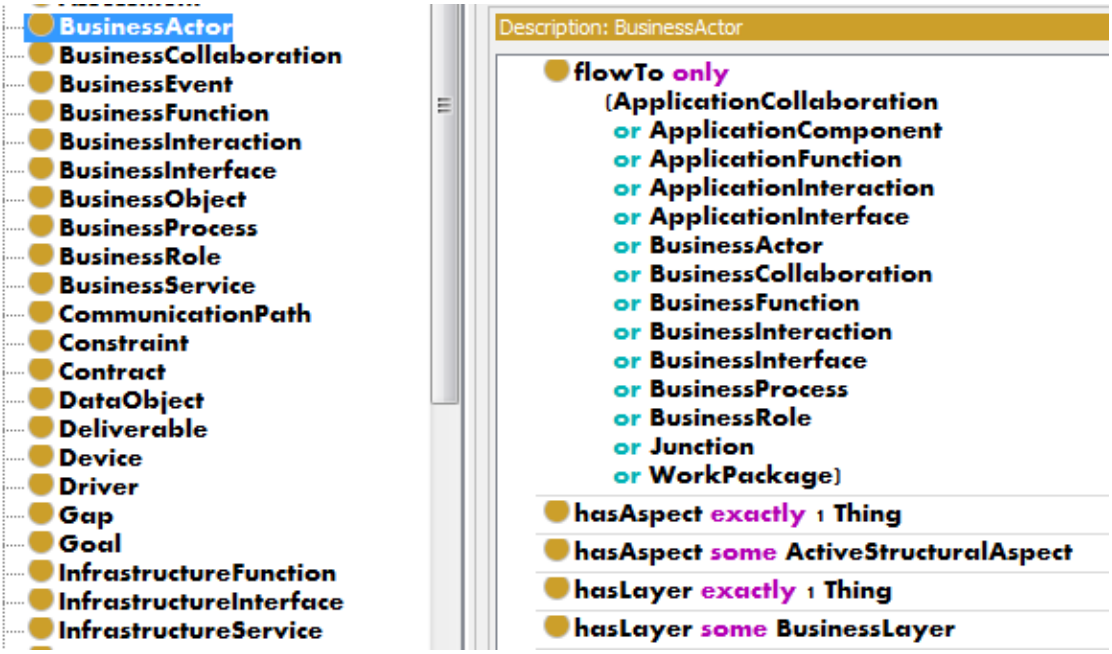

Figure 2 – Subset of ArchiMate ontology restrictions

4.2.Extension of the DIO through the DSO

The organizational stakeholders required modelling and analysing specific information about sensors since they play an important role in the structural monitoring and safety of civil engineering structures. However, the ArchiMate language lacks the expressiveness to capture the specifics of this domain. As a result, a specific description language needs to be defined. A simple sensors language was created for the demonstrations of the results. Figure 5 depicts a class diagram depicting the concepts of the DSO.

A mapping between this DSO and the DIO was created using an equivalentClass declaration. Sensors can be considered computational nodes, as they mix hardware and software for performing transformations. As such, the Sensor class in the Sensor DSO is considered equivalent to the Node class in the DIO. The GeoLocation class in the sensor DSO and the StructuralLocation class in the sensor DSO were considered equivalent to the Location class in the DIO, as Location is defined in ArchiMate as a point or extent in space, thus being more generic than the two concepts in the Sensor DSO. The Algorithm class in the Sensor DSO is considered equivalent to the ApplicationComponent class of the DIO. The Value class of the Sensor DSO is considered equivalent to the DataObject class of the DIO, the same happening with the AcquisitionRatePerYear class.

However, in order for this mapping to be correct, data properties needed to be added to DIO classes that are part of a mapping. For instance, the Node class of the DIO, along

with a data property restriction on the equivalence declaration, which declares that the equivalence exists when the hasType data property is present with the value "sensor" assigned to it. In this way, sensor nodes are disambiguated from other types of nodes. The mappings are the only exception where any changes are made to the DIO, in this case the addition of a data property, otherwise the principles behind this architecture would be defeated.

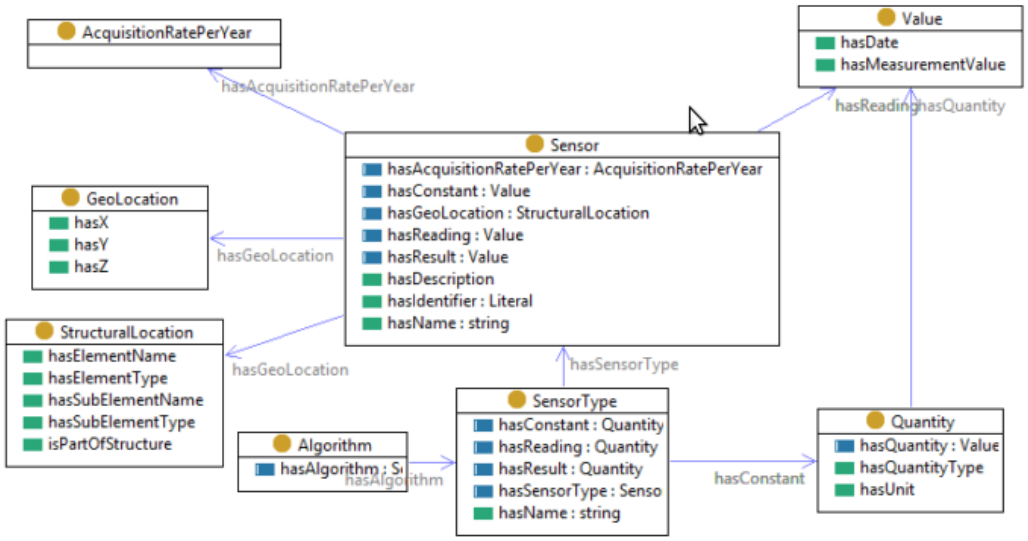

Figure 3 – Overview of the structure of the sensor ontology

With the transformation of the meta-model of the ArchiMate is possible to map this ontology with other ontologies, as sensors ontology, which was made by the mapping of concepts between the ontologies. With this mapping is possible to check how changes in one domain can cause in another domain.

In this sense, questions were made on ontology mapped, demonstrating the ability of analysis (Question 1) and other questions that span the domain ontology reaching ontology mapped (Question 2), showing also the traceability and validation. In question 2 was created a relationship derived (SuperObjectProperty) to model the dependencies between different elements, which was called dependsUp, this relationship uses the transitivity property of relations in ArchiMate involved in the creation, which means that a graph of dependencies can be created.

Question 1 - Which SensorType has the property hasReading with the time value?

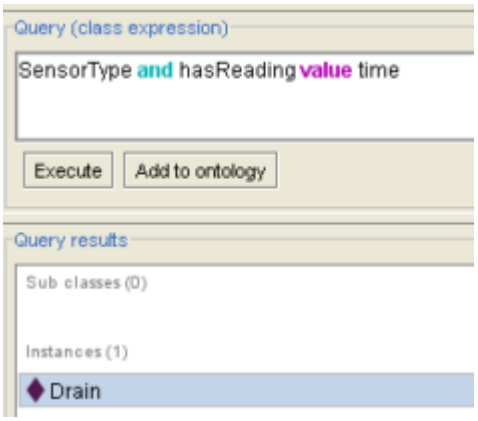

Figure 4 – Answer to question 1 about the DSO

Question 2 - Which ApplicationComponents have dependencies with any sensor with the property with the value hasSensorType Drain?

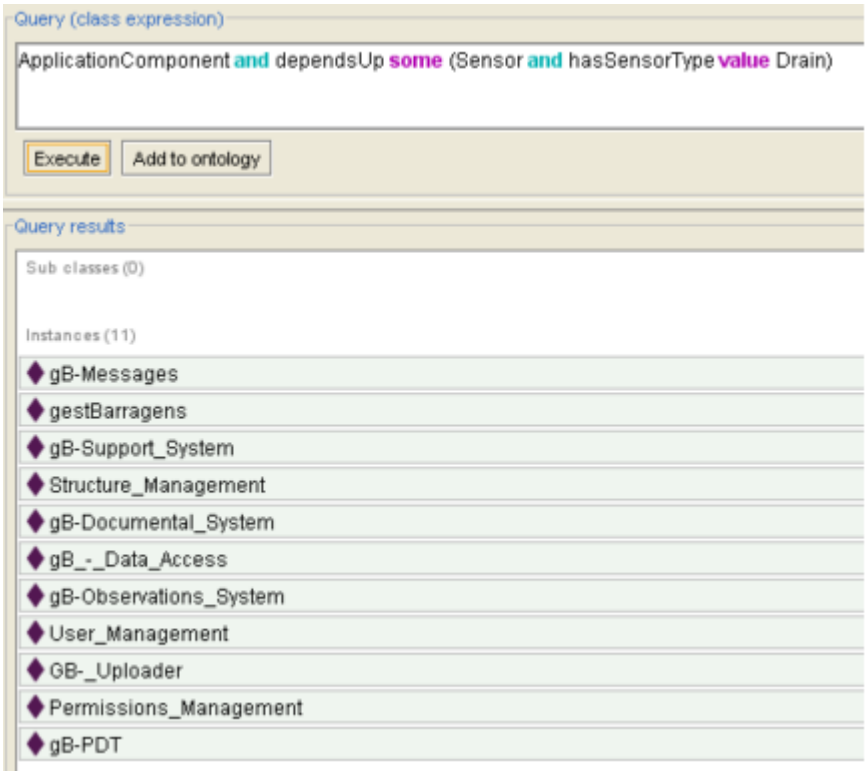

Figure 5 – Answer to question 2 about the DIO-DSO mapping

The validation of the questions answers were made by comparing the responses with the representation of the ontology of Sensors and the mapping done.

5. Conclusions

This work shows that the use of techniques of ontologies can assist the modeling of enterprise architectures through the analysis and validation of models, thus responding to the questions placed.

First of all, there was the transformation of the ArchiMate meta-model to ontology in order to support the analysis of models represented in this language.

Additionally, the representation of the ArchiMate meta-model and their respective models in OWL2 allows analyzing the content of the models with the rules. It is also possible to question the models.

Finally were demonstrated extension techniques of the ArchiMate to specific domains. The concepts and relations of a given vertical domain ontology is represented in a specific ontology, designated DSO. The concepts and relations of the DSO were mapped through an transformation ontology to the central ontology that specifies the DIO ArchiMate. Thus, it becomes possible to extend the language ArchiMate and integrate their concepts of the DSO. The mapping performed maintains consistency and traceability between concepts.

6. References

Binz, T., Leymann, F., Nowak, A., & Schumm, D. (2012). Improving the Manageability of Enterprise Topologies Through Segmentation, Graph Transformation, and Analysis Strategies. *IEEE Computer Society Conference Publishing Services.*

Buckl, S., Matthes, F., & Schewda, C. (2009). Classifying Enterprise Architecture Analysis. *Workshop on Enterprise Interoperability* (pp. 66-79). Springer.

Buckl, S., Matthes, F., & Schweda, C. (2010). Conceptual Models for Cross-cutting Aspects in Enterprise Architecture Modeling. *14th IEEE International Enterprise Distributed Object Computing Conference Workshops.*

Galster, M. (2011). Dependencies, traceability and consistency in software architecture:towards a view-based perspective. *Proceedings of the 5th European Conference on Software Architecture.*

Group, T. O. (2012). *Archimate 2.0 Specification.* The Open Group.

IEEE. (2011). ISO/IEC/IEEE 42010. *Systems and software engineering — Architecture description*.

Lankhorst, M. (2006). *Enterprise Architecture at Work.* Springer.

Romero, J., Jaen, J., & Vallecillo, A. (2009). Realizing Correspondences in Multi-Viewpoint Specifications. *Proceedings of the 2009 IEEE International Enterprise Distributed Object Computing Conference.*

Saat, J., Franke, U., Lagerström, R., & Ekstedt, M. (2010). Enterprise Architecture Meta Models for IT/Business Alignment Situations. *14th IEEE International Enterprise Distributed Object Computing Conference.*