ANALYSIS OF FEDERATED ENTERPRISE ARCHITECTURE MODELS

Complete Research[*]

Antunes, Gonçalo, INESC-ID, Rua Alves Redol 9, 1000-029 Lisboa, Portugal, goncalo.antunes@tecnico.ulisboa.pt

Barateiro, José, LNEC, National Laboratory for Civil Engineering, Avenida do Brasil 101, 1700-066 Lisboa, Portugal and INESC-ID, Rua Alves Redol 9, 1000-029 Lisboa, Portugal, jbarateiro@lnec.pt

Caetano, Artur, IST, University of Lisbon, Avenida Rovisco Pais 1, 1049-001 Lisboa, Portugal and INESC-ID, Rua Alves Redol 9, 1000-029 Lisboa, Portugal, artur.caetano@tecnico.ulisboa.pt

Borbinha, José, IST, University of Lisbon, Avenida Rovisco Pais 1, 1049-001 Lisboa, Portugal and INESC-ID, Rua Alves Redol 9, 1000-029 Lisboa, Portugal, jose.borbinha@tecnico.ulisboa.pt

Abstract

Enterprise architecture models support decision-making as they help organizations to understand, communicate and analyse how business processes are performed, the goals they achieve, the information they use, as well as the applications that realize the business, and the supporting technological infrastructure. The integrated analysis of these models is not straightforward because it cross-cuts different domains that are described using heterogeneous concepts. This paper explores the application of ontologies to analyse enterprise architecture models. Ontologies represent knowledge that can be analysed using computational inference. The contributions of this paper are (1) the specification of multiple enterprise architecture models as ontological schemas, (2) the integration of ontological schemas, and (3) the analysis of the integrated models. The solution artefact consists of a federated model specified as a set of ontological schemas described in OWL-DL that integrates multiple enterprise models and assists their analysis using computational inference. The paper demonstrates the application of the federated model to the ArchiMate language as a means to assess the compliance of business requirements in a civil engineering scenario.

Keywords: enterprise architecture, ontology, model analysis, model integration.

1 Introduction

Enterprise architecture model analysis focusses on the application of techniques that examine the model's artefacts, their properties and dependencies, and generate information that can be used to assess, transform or redesign the organizational systems(Bucher et al., 2006; Johnson, Lagerstrom, et al., 2007). Model analysis can be seen as the "application of property assessment to enterprise architecture models" (Johnson, Lagerstrom, et al., 2007), with the goal of observing a system's functional and non-functional qualities (Bertolino et al., 2013).

[*] This work was partially funded by FCT, Fundação para a Ciência e a Tecnologia through projects UID/CEC/50021/2013 and DataStorm EXCL/EEI- ESS/0257/2012.

Model analysis requires that models contain the necessary information that drives the overall analysis process (Narman et al., 2007). One approach to model analysis is defining a single and unified meta-model that grounds the instantiation and design of models and views (Fischer et al., 2007). These "holistic" meta-models often focus on aligning multiple concepts and constrain the design and analysis of the architectural description to the structure and semantics of the meta-model. Designing a unified meta-model raises problems due to its broad scope and lack of specificity, and also to the problems related to extending the meta-model to address additional concepts while keeping it consistent. Some authors argue that this approach is not adequate to capture the diversity of domain-specific aspects since its high-level of abstraction often lacks the expressiveness to meet all design requirements (Buckl, Buschle, et al., 2011; Saat et al., 2010; Sousa et al., 2006). Other techniques rely on designing and adding a set of properties to the model's artefacts as a means to analyse the models (Bucher et al., 2006; Buckl et al., 2009; Hamalainen, 2008; Niemann, 2005). Instead of using a single meta-model, a system can also be conceptualized by a set of meta-models to address the specificity of its domains (Fischer et al., 2007; Zivkovic et al., 2007) and the concerns of its stakeholders (Kurpjuweit et al., 2007). Domain-specificity is also acknowledged by the application of situational modelling to enterprise architecture management and design, that facilitate the selection of a suitable approach according to the constraints of the organizational context and the goals of architecture project (Buckl et al., 2010; Buckl, Schweda, et al., 2010; Caetano, Pereira, et al., 2011; Caetano, Silva, et al., 2009; Saat et al., 2010).

An ontology is a "formal, explicit specification of a shared conceptualisation" (Studer et al., 1998). Ontologies enable representing the aspects of a conceptual model, such as its conceptual schema and information base (Olivé, 2007). Inference can be used to derive logical conclusions from the ontological facts and support model analysis. Ontologies also provide the mechanisms to integrate different models or schemas through the definition of rules that relate the concepts at structural or semantic level.

This project followed the Design Science Research method (Peffers et al., 2008). The DSR method involves six steps: (*i*) problem identification; (*ii*) definition of the solution objectives; (*iii*) design and development of the solution artefact; (*iv)* demonstration; (*v*) evaluation; and (*vi*) communication.

Step (*i*) identified schema integration and logical inference as factors that may contribute to enterprise architecture model analysis. The main contribution of this paper is therefore exploring *how ontologies can support enterprise architecture model analysis* and not comparing ontologies with other techniques that can be used to the same end. Thus, this paper examines the following hypothesis:

H1. Ontologies support the computational representation of enterprise architecture schemas.

H2. Ontologies support enterprise architecture schema integration.

H3. Logical inference enables the computational analysis of enterprise architecture schemas according to the semantic and syntactic rules represented in the ontologies.

Step (*ii*) involved defining the requirements of a solution artefact according to the hypothesis. Step (*iii*) is the design of the solution artefact. Steps (*iv*) and (*v*) demonstrate and evaluate the solution artefact through a case study that assesses the compliance of an architectural description against a set of requirements. This paper is a direct outcome of step *(vi)*.

The remainder of this paper is structured as follows. Section 2 provides an overview on enterprise architecture model analysis and ontologies. Section 3 describes the solution artefact for the analysis of federated enterprise architecture models. Section 4 demonstrates the application of the solution artefact and section 5 concludes the paper.

2 Fundamental concepts

This section reviews the main concepts and definitions related to enterprise architecture model analysis (section 2.1) and ontologies (section 2.2).

2.1 Enterprise architecture model analysis

According to Hamalainen (2008), enterprise architecture model analysis assists the continuous enterprise architecture program by providing information to support the planning, improvement, and management processes. It also supports the governance, optimization, re-engineering and decision-making processes of an organization (Caetano, Assis, et al., 2012; Hamalainen, 2008; Johnson and Ekstedt, 2007; Lankhorst, 2005). Niemann (2005) classifies model analysis according to the following categories: *dependency analysis* relates enterprise architecture artefacts to derive direct and indirect dependencies between the artefacts; *coverage analysis* detects redundancies and gaps between artefacts belonging to two or more EA layers; *interface analysis* assesses how the interfaces of artefacts relate usually with the goal of determining the degree of coupling and cohesion; *heterogeneity analysis* identifies elements that need re-factoring as means to homogenize the overall architecture description; *complexity analysis* measures the complexity of the model usually using variants of Kolmogorov complexity or algorithmic information metrics to determine the computability resources needed to specify the model (Burgin, 2009); *compliance analysis* determines if the artefacts or the overall architecture description meet policies, rules and requirements; *cost analysis* calculates the costs of an artefact (e.g creation, maintenance) or costs pertaining to the architecture description; and *benefit analysis* determines the contribution of an artefact to the overall goals of the organization as described in architecture.

One avenue to realize these analytical techniques is to enrich the enterprise architecture artefacts to prepare the models for analysis. Other avenue is using the model's structural and semantic properties as specified in its meta-model, or as implicitly inferred from its objects. Johnson, Ekstedt, et al. (2004) propose adding non-functional attributes to enterprise architecture models through architecture theory diagrams (ATD). These diagrams interrelate attributes through composition, correlation, and casual relationships. The ATD is then populated with measures that support the calculation of the model's non-functional attributes. Extended influence diagrams (EID) (Johnson, Lagerstrom, et al., 2007) and probabilistic relational models (PRM) (Lagerstrom et al., 2010) are other analysis techniques. EID are used to model goals and decision alternatives, thus providing support for decision making. EID use probabilistic inference to support the representation of uncertainty when computing the values of attributes. PRM extend entity relationship models, and support model analysis under uncertainty. Davoudi et al. (2012) propose using the analytical hierarchical process (AHP) to prioritize and select architectural scenarios according to the non-functional requirements under analysis. Franke et al. (2009) use fault tree analysis (FTA), an extended Bayesian network to analyse dependencies related to reliability and reusability qualities. Caetano, Assis, et al. (2012) use model mapping and transformation to analyse the compliance of process models against a set of actor coordination patterns. Boer et al. (2005) use a XML schema to encode an enterprise architecture meta-model that specifies the structure and dynamics of enterprise architecture models. This approach analyses the structure of a model in terms of its cardinality, class specialization, and concept relationships. The dynamic analysis uses scenarios that encode state-based actions with XML and RML rules to simulate the behaviour of the architecture.

2.2 Ontologies

The term *ontology* originates from the Greek language and it compounds *ontos* (being) with *logos* (word). From a philosophical perspective, ontology is a "systematic explanation of existence" (Gomez-Perez et al., 1999). In computer science, an ontology is defined as a "formal, explicit specification of a shared conceptualisation" (Studer et al., 1998), being Studer's definition, which is based on Gruber's Gruber (1993) and Borst's Borst (1997), likely the most commonly used. According to Guarino et al. (2009) and Genesereth et al. (1987), *conceptualisation* refers to an "abstract, simplified view of the world", containing "the objects, concepts, and other entities that are assumed to exist in some area of interest and the relationships that hold among them". Studer et al. (1998) relate *explicit* to the definition of the "type of concepts used, and the constraints on their use" while *formal* to the fact that the conceptualisation "should

be machine readable" Finally, *shared* means that the ontology "captures consensual knowledge".

Description logics (DL) are "a family of logic-based knowledge representation languages suitable for the representation of ontologies", which can be seen as "a decidable fragment of first-order logic" (Vaculin, 2009). DL describe domains in terms of *concepts*, *roles*, and *individuals*. Roles and concepts are related using logical statements named *axioms*. Axioms can be of two kinds: terminological and assertional. *Terminological axioms* describe concepts and properties of the concepts, while *assertional axioms* are statements about the individuals belonging to the concepts that are compatible with the terminological axioms. In DL, a *T-Box* is any finite set of terminological axioms and an *A-Box* is a finite set of assertional axioms. Together, the set of all T-Box and A-Box statements define the *knowledge base*.

Different varieties of DL exist with differing degrees of expressiveness. The base DL is the Attributive Language with Complements ($\mathcal{ALC}$), which allows expressing concepts along with a set of concept relationships: union, intersection, complement, universal restriction, and existential restriction. The expressiveness of $\mathcal{ALC}$ can be augmented with relationships such as transitive and inverse roles, role hierarchy, cardinality restrictions, qualified cardinality restrictions, concrete domains, and enumerated classes (Lemaignan, 2012).

DL supports five different types of reasoning (Areces, 2000): subsumption, instance checking, relation checking, concept consistency, and knowledge base consistency. *Subsumption* organizes concepts in a hierarchy and finds the most specific super-class for each class. *Instance checking* verifies if a given individual is an instance of a concept. *Relation checking* verifies if and how two individuals relate to each other. *Concept consistency* verifies if there are no contradictions between the definitions or the chain of definitions of a concept. Finally, *knowledge base consistency* determines whether the information contained in the knowledge base contains any contradictions.

3 Application of ontologies to the analysis of architecture models

The hypothesis behind this project aim to test the application of ontologies as a means to support the computational analysis of heterogeneous enterprise architecture models. This goal raises the following requirements that need to be met by a solution artefact:

R1. The solution must represent the conceptual schemas and information bases that model the enterprise architecture domains using ontologies formalized with $\mathcal{ALC}$ description logics.

R2. The solution must provide means to integrate the conceptual schemas.

R3. The solution must support model analysis using computational inference.

The next three sections describe the realization of each of the solution requirements, namely how schemas are represented (section 3.1), how schemas are integrated (section 3.2) and, how reasoning is used to analyse the integrated schemas (section 3.3).

3.1 Schema representation

Requirement *R1* states that an enterprise architecture model is specified using a conceptual schema that defines the entity and relationship types of the domain, and that the instances of entities and relationships of the domain are classified according to the types defined in the conceptual schema (Olivé, 2007). Ontologies, in particular those formalized using $\mathcal{ALC}$ description logics, are suitable to specify the conceptual schemas and the information base (Breitman et al., 2007). Using schemas to support the definition of viewpoints and the generation of views accordingly to the viewpoints is also a desirable requirement. Views and viewpoints facilitate the separation of concerns by isolating certain aspects of the architecture according to the requirements of its stakeholders (ISO/IEC/IEEE, 2011). Ontologies enable to apply computational inference to transform and analyse the conceptual schemas. As a result, a viewpoint can be defined as a set of ontological axioms that are used to infer a view from the schema. For instance, ontological axioms can define a viewpoint that filters the model's schema according to a single entity type, such as "select all entities of type *business process*"), or to generate more complex viewpoints such

as "select all entities of type *business process* that relate to at least two entities of type *business object* through the *use* relationship".

3.2 Schema integration

Requirement *R2* derives from the observation that enterprise architecture models often cross-cut multiple domains. Using a unified meta-model that encompasses the concepts included in the multiple domains is an option to address this issue. The main advantage is that the meta-model can be designed with consistency in mind, but at the expense of using a high level of abstraction. However, the level of abstraction may not suit the goals of the analysis or the meta-model may not even contain the domain-specific types required for a modelling task. This limitation can be addressed through the federated or integrated usage of multiple models or domain-specific languages (Bjeković et al., 2013; Fischer et al., 2007; Zivkovic et al., 2007). Ontologies have been used to promote the interoperability between different domains, often through the diagnosis and integration of schemas and the identification of matches and mismatches between classes and individuals of different ontologies (Pinto et al., 1999; Uschold et al., 1996). Thus, requirement *R2* uses model integration as a mean to address the domain-specificity of enterprise architecture models. However, the integration of models should be done in such a way that existing schemas are not modified, meaning that introducing domain-specific aspects to the model must not interfere with concepts already defined.

To do so, we propose using a federated model landscape to facilitate the integration of the different domains of the architecture. The federated landscape contains two core elements: the *upper ontology* (UO) and the *domain-specific ontology* (DSO). The UO is a schema or meta-model that represents a set of core entity and relationship types. The UO is domain-independent in the context of enterprise architecture, i.e. it does not address domain-dependent concerns. The UO should also be defined in such a way that its types are consistent and non-redundant. A DSO is a domain-specific language that addresses a particular set of well-defined and bounded concerns. A DSO should only contain the minimum set of entity and relationship types that are required to satisfy the needs of the stakeholders behind its concerns. Thus, the UO and the set of DSO should be designed in such a way that their underlying concerns are clearly separated (Tarr et al., 1999).

Each DSO can be integrated with the UO, thereby creating a federated landscape of models. The relationships between a DSO and the UO derive from the analysis requirements, meaning that these UO-DSO relationships must be traceable to the concerns of the stakeholders. In terms of solution, this implies integrating the upper ontology with each domain-specific ontology using an *integration map*. An integration map is itself a conceptual schema that specifies the structure and semantics behind the relationships between the UO and a DSO. The map may consist of a a set of 1:1 relationships between a subset of the UO types and the DSO types. However, a relationship may also be defined by a set of axioms or inference results. In either case, the map will likely not lead to isomorphic relationships between all UO and DSO types because a DSO is designed as domain-specific specialization or extension of a part of the UO. Semantic mismatches between the UO and a DSO may arise due to differences in coverage, granularity and perspective between the schemas, and may lead to construct incompleteness, redundancy, excess or overload (Wand et al., 1993). Depending on the goal of the project, these modelling deficiencies can be assessed and addressed using specific techniques (Breitman et al., 2007; Fettke et al., 2003). Nevertheless, the federated model that is presented in this paper is independent of the method that is used to deal with the modelling deficiencies identified during UO-DSO mapping.

Figure 1 depicts a rich diagram with the mapping relationships between the UO and a set of DSO. A map between each (UO, DSO) pair contains all structural and semantic relationships between the UO and DSO schemas. It may directly relate each DSO type to a UO type using *subclassing* or type *equivalence*. A type may also be mapped according to specific properties of a class or individual or as the result of inference.

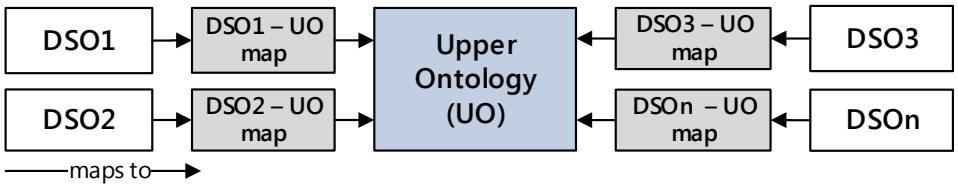

Figure 1. The ontology-based federated model for representing and integrating multiple schemas.

	Model analysis						
	Dependency	Coverage	Interface	Heterogeneity	Complexity	Compliance	Cost/benefit
Subsumption	❍	❍	❍	●	❍	❍	❍
Instance checking	❍	❍	❍	●	❍	❍	❍
Relation checking	◗	●	●	❍	◗	❍	◗
Concept consistency	❍	❍	❍	❍	❍	●	❍
KB consistency	❍	❍	❍	❍	❍	●	❍

Table 1. Application of reasoning to model analysis. ● means full coverage, ◗ partial coverage and ❍ no coverage.

3.3 Model analysis

Requirement *R3* takes advantage of computational inference to perform automated model analysis. There are three categories of analysis regarding the number and type of schemas that are used, namely:

- *Intra-schema inference* analyses types within a single schema. If the types of the schema are layered, then inference may target a single layer (intra-layer reasoning) or several layers (inter-layer reasoning). This type of inference can be specialized as *intra-UO inference* when the types are contained in the upper ontology, or to *intra-DSO inference* when the types are contained in a single DSO.
- *Inter-schema inference* analyses types from a pair of schemas (S1, S2). This inference becomes *UO-DSO inference* when the analyses use types from the DSO and the UO schemas and requires a (UO, DSO) integration map. It is *DSO-DSO inference* when the schemas are a pair of DSO.
- *Chained inference* combines multiple applications of intra- and inter-schema inference to analyse a federation of schemas.

Description logics uses inference to provide five different types of reasoning (Areces, 2000): subsumption, instance checking, relation checking, concept consistency, and knowledge base consistency (cf. section 2.2). These can be used to analyse the entity and relationships types specified in the conceptual schema as well as the individual instances of these types. Table 1 shows the relationship between the different types of reasoning and the seven categories of model analysis proposed by Niemann (2005) and summarized in section 2. Full coverage indicates that the reasoning technique directly supports the model analysis type, while partially coverage means that the reasoning technique can contribute to model analysis but will require a combination with other techniques.

Subsumption and *instance checking* support the analysis of the *heterogeneity* of a model, as it involves classifying types and its individuals in order to refactor the model. *Relation checking* supports several types of model analysis since it determines how entity types relate to each other through a chain of relationship types. Thus it supports *dependency analysis* as it involves checking whether two elements are related with each other and determining the nature of the relationship. It also supports analysing the *coverage* of a model, i.e. how the model's artefacts intersect the architecture layers. It also plays a role in *interface* analysis since it enables assessing the degrees of cohesion and coupling of an artefact. Finally, relation checking partially aids the computation of metrics required for *complexity* analysis and *cost/benefit* analysis. *Concept consistency* and *knowledge base consistency* can aid *compliance* assessment.

Concept consistency can be used to verify the existence of contradictions between the definitions of a type. This can be used to determine whether rules or requirements are being met. Knowledge base consistency checking can be used to verify whether the definition of an individual entity or relationship complies with the definition of the corresponding type, i.e. whether a model is valid against the specification of a meta-model.

Nevertheless, reasoning is not able to fully support all categories of model analysis, as evidenced in Table 1. In particular, the application of reasoning to dependency analysis is limited to the relationships between different types since it is not possible to analyse relationships between type properties. Reasoning is also not a good candidate to perform the computation of metrics required for complexity analysis and cost/benefit analysis. Thus, description logics based reasoning requires to be complemented with other analytical techniques. Moreover, the suitability of using reasoning also needs to be studied in terms of performance and scalability, especially when analysing large enterprise architecture models with a dense graph of relationships between the entities.

4 Demonstration

This section describes the application of the federated model described in the previous section to a specific scenario. The application includes the instantiation of the UO, the instantiation of one DSO, the integration of the DSO with the UO, and examples of intra- and inter-schema inference to analyse the integrated models.

The scenario concerns part of a long-running business process that monitors the structural behaviour of large concrete dams during their lifestyle. These dams are part of larger systems that often include hydroelectric power plants and the reservoirs. The results of this monitoring process are used to assess the structural state, which are then used to plan measures that aim to ensure the structural integrity of the dam and thereby minimizing the overall risk of accident. The safety assessment process is a responsibility of each dam owner, supported by the National Laboratory of Civil Engineering and must legally comply with a number of directives and requirements. Each dam is monitored using a vast array of different types of sensors that measure different types of physical and chemical phenomena along time. Some of these sensors are deeply embedded in the foundations and structure of the dam during the construction phase, others are installed after construction, while other sensors may be used temporarily to observe a particular feature of the dam.

This organization is legally required to maintain an information system to manage the information produced by the monitoring process. One of the activities of the monitoring process is *data acquisition*. Its main purpose is to acquire the raw data generated by sensors. The activity then performs an initial validation of the data, and communicates it to the information system that manages the monitoring data. As a best practice, this organisation has modelled these processes, along with its supporting applications and technological infra-structure. For that purpose, it has produced an enterprise architecture representation using the ArchiMate modelling language (The Open Group, 2012). Figure 2 shows an ArchiMate model that relates to the monitoring process. This technological infrastructure view describes the deployment of an application server at infrastructure level. This application server is where the monitoring management information system is deployed. It interfaces with a database server that stores the monitoring data, and with the producer node that captures the monitoring data using sensors, which function automatically or manually. The system is managed via a web interface.

In order to analyse this model, we first specified the ArchiMate meta-model, represented in Figure 3, as an upper ontology. For the next step we used an automated translation script to convert the XML representation of the ArchiMate model into an ontological schema in the OWL-DL language. The XML representation of the model was obtained through the Archi modelling tool. As a result, we generated an ontology in OWL-DL populated with individuals for each entity and relationships of the ArchiMate model that was classified according to the types of the ArchiMate meta-model.

However, the ArchiMate language is designed to consistently conceptualize the core entity and

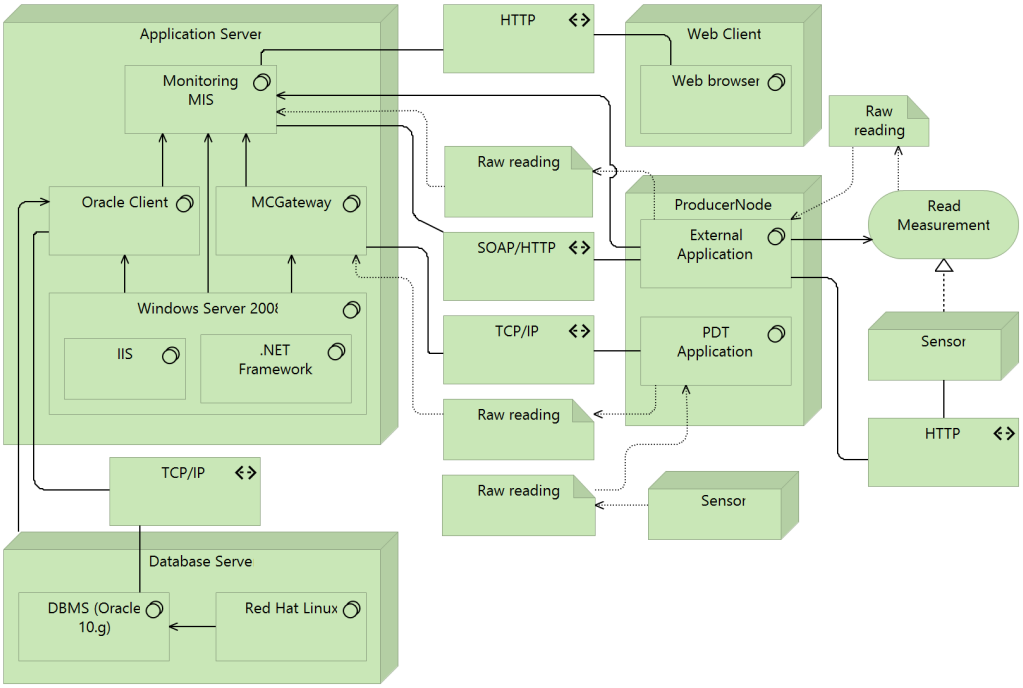

Figure 2. ArchiMate model view depicting the technological infrastructure dependencies of an Application Server.

relationship types that relate the business, application and technological domains but at a high level of detail. As such, the concepts pertaining to the specific sensor domain are not to be found on such a language. Thus, we designed a domain-specific ontology to extend the expressiveness of the ArchiMate language to the sensor domain without modifying the underlying upper ontology. Sensors measure values that can be processed and used in the predictive analysis of structural behaviour. A sensor measures physical values and make use of specific calibrations and algorithms to generate the observation data. For this reason there are several types of specialized sensors. To capture this domain we designed a sensor DSO according to the requirements of the organization. The UML class diagram in Figure 4 shows the conceptual model of the sensor domain.

Table 2 shows the mapping of the entity types of the sensor DSO to the upper ontology types. The map is an OWL-DL ontology that relates the DSO types with the UO types using the *subClassOf* construct. The subclass relations provide the necessary conditions for determining whether class membership. This means that if the class description C1 is defined as a subclass of class description C2, then the set of individuals in the class extension of C1 should be a subset of the set of individuals in the class extension of C2 (W3C, 2012). Besides the mappings at the class level, additional mappings need to be created at the instance level. In this case, the Sensor instances defined in the respective DSO need to be mapped to their counterpart in the UO. Given that the UO is representing both automatic (i.e., sensors that provide an infrastructure service) and manual sensors, the integration at the instance level is required for adding this information to the one present in the DSO. The built in owl property *owl:sameAs* can be used for performing this mapping.

One of the analysis questions asked by the scenario stakeholders was the following: “can the existing monitoring system be used to acquire monitoring data from bridges?” To understand the requirements behind this question we decomposed its goals until we could assess whether the underlying requirements could be analysed using the existing models. Figure 5 shows a partial goal decomposition diagram modelled with the *i** language (Yu, 1997). The goal model is limited to the requirements related to the extensometer sensor for the sake of simplicity. The structural safety of a bridge requires three models:

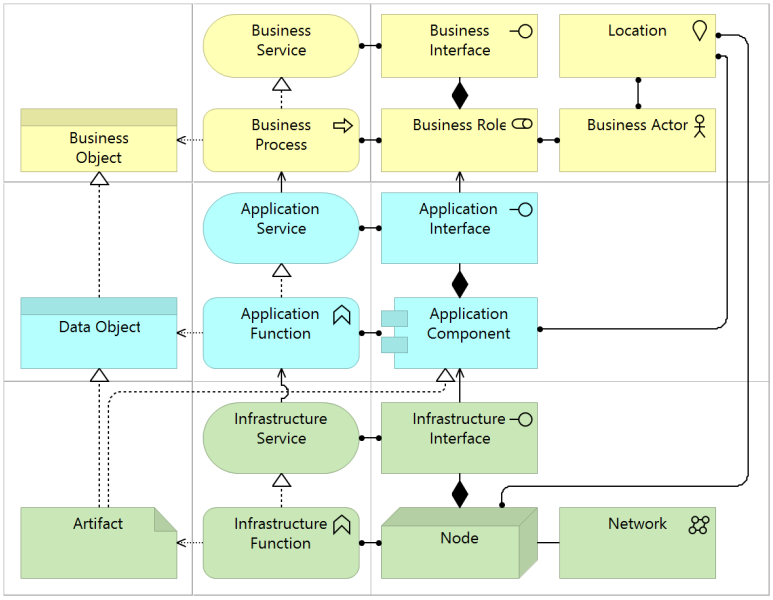

Figure 3. Simplified ArchiMate meta-model (adapted from (Iacob et al., 2012; The Open Group, 2012)).

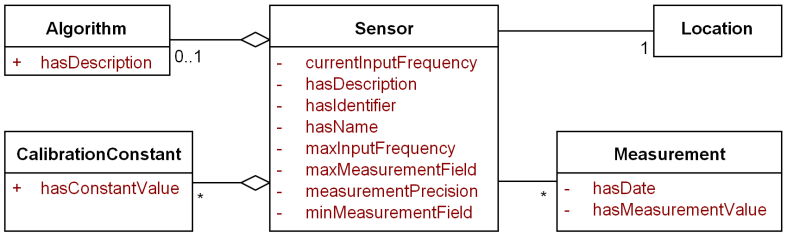

Figure 4. UML class diagram of the sensor DSO.

Sensor DSO	ArchiMate UO
Sensor	Node
Measurement	Data Object
Constant	Data Object
Location	Location
Algorithm	Application Component

Table 2. Entity type mapping between the sensor domain-specific ontology and the upper ontology.

a structural model, a theoretical model, and a statistical model. The elaboration of the structural model requires up-to-date data acquired from the monitoring equipment installed in the structure. As it was already described above, monitoring a dam requires the acquisition, validation, storage and processing of the sensor data. The stakeholders' question refers only to the aspect of data acquisition. For example, the monitoring of a bridge requires a high frequency of data acquisition, which therefore needs to be fully automated (requirement *DA1*). Morever, the acquisition process needs to fulfil three additional soft-goals: (*DA2*) the sensor must have a sampling rate between 500 Hz and 1 kHz, (*DA3*) it must measure extensions in the range of -20 cm to +20 cm (note that the structure of a bridge needs to be more elastic than that of a dam and thus requires a larger interval of measurement), and (*DA4*) the measuring precision needs to be at most 0.1 mm. Requirements DA1-4 were formalized in OWL-DL using the four axioms described next. The first axiom can be used to verify the automation requirement DA1, through performing a simple DL

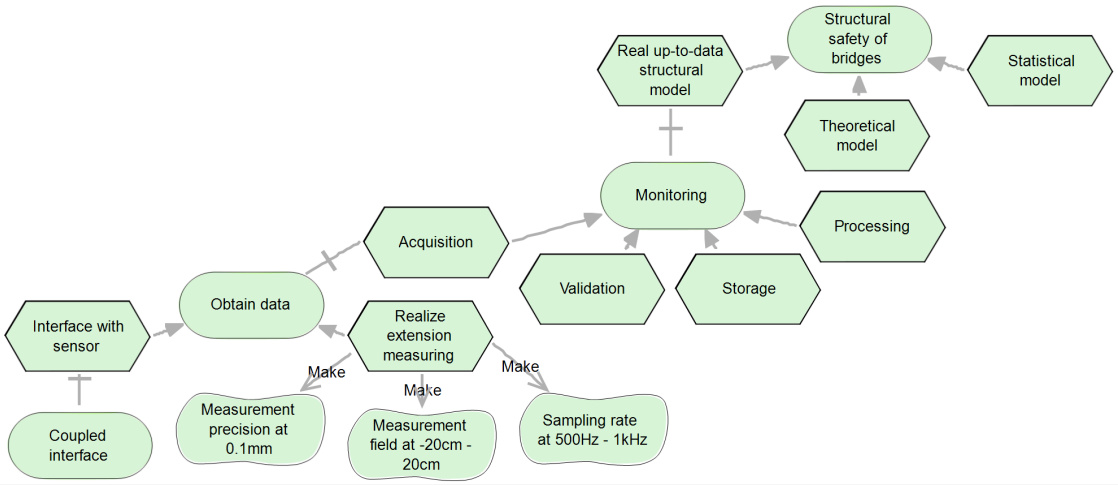

Figure 5. Partial i goal decomposition diagram.*

query for obtaining this information, as it cannot be defined as a constraint on the model, due to OWL's open world assumption (i.e., manual sensors would have to be modelled explicitly as not providing an infrastructure service, which would involve changing the ArchiMate axioms). Regarding the soft-goals identified in the goal decomposition diagram, the constraints are specified as the next three ontological axioms and translate the requirements *DA2-4*.

- **DA1: coupledInterfaces** = **Sensor** (and *realizes* some **InfrastructureService**).
- **DA2: measurementPrecision** = **Requirement** and *realizedBy* only (**Sensor** and *measurementPrecisionInMm* only float[<= 0.1])
- **DA3: measurementField** = **Requirement** and *realizedBy* only (**Sensor** and *maxMeasurementFieldInCm* only float[>= 20]) and *realizedBy* only (*Sensor* and *minMeasurementFieldInCm* only float[<= -20.0])
- **DA4: samplingRate** = **Requirement** and *realizedBy* only (**Sensor** and *maxInputFrequencyInHz* only float[<= 1000.0] and float[>= 500.0])

Figure 6 shows a partial result of this analysis, in which the reasoner detected three inconsistencies in the ontology. The two first inconsistencies (explanation 1 and 2) arise because *Extensometer1-01* does not comply with the measurement requirement *DA3* as it performs measurements within the range [-5 cm, 5 cm]. The fourth inconsistency (explanation 3) states that *Extensometer1-01* has a sampling rate of 10 Hz, outside the sampling rate interval specified in *DA2*.

This section explained the application of the federated model to the integrated analysis of a technological infrastructure view along with a domain-specific language. The technological infrastructure view was expressed with the ArchiMate language as part of an enterprise architecture representation that models a sensor monitoring process in a national laboratory to assess the structural integrity of dams. The original ArchiMate models were extended with a domain-specific language that describes the sensor domain with the detail that is required to model the monitoring process. The demonstration portrays (1) the use of the ArchiMate language as an upper ontology, (2) the specification of the sensor domain with a domain-specific ontology, (3) the integration of these two schemas using a map between the types of the UO and the DSO, (4) the application of inter- and intra-schema inference to analyse the entities and relationships of the UO, the DSO, and of the integrated UO and DSO, and finally (5) the application of

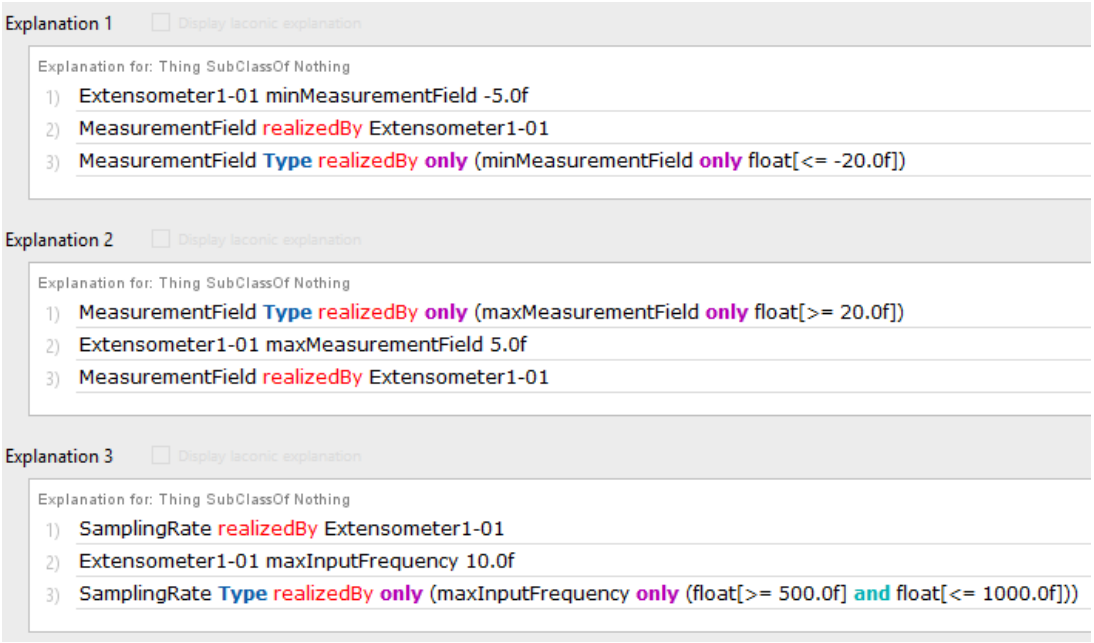

Figure 6. Inconsistencies detected by the reasoner while checking the compliance of the ontology against the DA2-4 constraints.

inference to assess the compliance of the model against a set of goals and requirements.

5 Conclusions

Enterprise architecture models often cross-cut different and heterogeneous domains, such as goals, requirements, business processes, indicators, people, systems, services, and technology. These domains are observed from different viewpoints that produce the views that the stakeholders use to communicate, understand and analyse the system. One approach to analyse these multiple domain is using a single enterprise architecture schema or meta-model that conceptualizes all domains in a unified way. This approach enables the meta-model to be consistently and economically designed but at the expense of domain-specificity since the concepts are specified at a high-level of abstraction. As a result, a unified meta-model can be used to support model analysis from a high-level of abstraction which is essential to assess model consistency and the overall alignment between the concepts. However, this approach has limitations when it comes to dealing with the specific domains, either in terms of their analysis or the specification of viewpoints. In this setting, this paper explores how to use a federated set of models that is able to address domain specific concerns. The goal is therefore to be able to analyse the resulting integrated model landscape. In particular, this paper explores how to use ontologies to specify the enterprise architecture model landscape, with the goal of supporting model analysis.

The solution artefact is a federated set of ontological schemas that represent the domain-independent and domain-specific meta-models (the conceptual schemas), and the instances of those meta-models (the information bases). The upper ontology (UO) specifies the domain-independent meta-model through a core set of entities and relationship types. Each domain-dependent meta-model is represented as a domain-specific ontology (DSO) that specifies its types. The DSO types are independent of the UO types. The model landscape is generated by integrating the schemas (i.e integrating a DSO with the UO or integrating several DSO). Each integration between a pair of schemas is itself an ontological schema that specifies how the entity and relationship types relate using ontological axioms. The way the schema types are mapped depends on the requirements of the analysis. This means the mapping is situational, in the

sense that there is no universal mapping between a pair of schemas. Accordingly, each schema integration map results from a specific set of concerns defined by the system's stakeholders.

The federate set of ontological schemas was formalized with $\mathcal{ALC}$ description logics using the OWL-DL language. This approach enables using reasoners to perform logical inference to analyse the models. The analysis may target a single domain (i.e. the UO or one of the DSO), or a combination of several domains. Different types of analysis are also possible, although inference primarily assist analysing how the concepts relate and how the concepts are classified. This can be used to assess the compliance of a model against a meta-model or against a set of rules or constraints. As such, logics-based inference is not a good candidate to support quantitative model analysis.

The solution artefact was used to determine whether the current technological infrastructure that supports a data monitoring process could also be used to support other monitoring process that needs to fulfil a set of specific requirements. The demonstration involved specifying an upper ontology, a domain-specific ontology, their integration, the specification of the compliance requirements, and, finally, the analysis of the integrated models.

The main contribution of this paper is showing the benefits and limitations of using ontology-based techniques to analyse multiple enterprise architecture models. In particular, this paper shows how to specify and integrate multiple models, and how to use computational inference to support model analysis. Although the application of OWL-DL ontologies and inference to the specification and analysis of enterprise architecture models is neither a complete nor a sufficient solution, it can however bring value to the enterprise architecture community of practice as it does contribute to support the federated specification of diverse enterprise architecture domains along with their situational analysis.

Our current work focuses on the application of this approach to the integration of specific enterprise architecture domains, namely value, business requirements, business services, and business processes. The goal is to identify and separate the concerns pertaining to each domain, refactor the domains, and then federate the refactored models using ontological schemas. For this purpose, related approaches on ontology modularisation and integration are being considered. The primary goal is enabling the specification of viewpoints and assessing compliance against requirements. Other research avenue relates to evaluate how ontologies compare with other techniques in order to design a situated method that selects and combines the most suitable techniques depending on the analysis concerns. Finally, ontologies must also be evaluated in terms of performance and scalability when applied to large-scale or densely connected enterprise models.

References

Areces, C. (2000). "Logic Engineering. The Case of Description and Hybrid Logics." PhD thesis. Institute for Logic, Language and Computation, University of Amsterdam.

Bertolino, A., P. Inverardi, and H. Muccini (2013). "Software architecture-based analysis and testing: a look into achievements and future challenges." *Computing* 95, 633–648.

Bjeković, M. and H. A. Proper (2013). "Challenges Of Modelling Landscapes: Pragmatics Swept Under the Carpet?" In: *Proceedings of Third International Symposium Business Modeling and Software Design, Lecture Notes in Business Information Processing (Book 173)*. Springer, pp. 11–22.

Boer, F. de, M. Bonsangue, J. Jacob, A. Stam, and L. der Torre (2005). "Enterprise Architecture Analysis with XML." In: *Proceedings of the 38th Hawaii International Conference on System Sciences (HICSS)*.

Borst, W. N. (1997). "Construction of Engineering Ontologies." PhD thesis. University of Twente, Enschede.

Breitman, K., M. A. Casanova, and W. Truszkowski (2007). *Semantic web: concepts, technologies and applications*. Springer.

Bucher, T., R. Fisher, S. Kurpjuweit, and R. Winter (2006). "Enterprise architecture analysis and application. An exploratory study." In: *Proceedings of the 1st Workshop on Trends in Enterprise Architecture Research (TEAR)*.

Buckl, S., M. Buschle, P. Johnson, F. Matthes, and C. M. Schweda (2011). "A meta-language for Enterprise Architecture analysis." In: *16th International Conference on Exploring Modeling Methods for Systems Analysis and Design (EMMSAD)*.

Buckl, S., F. Matthes, and C. M. Schweda (2010). "Conceptual models for cross-cutting aspects in Enterprise Architecture modeling." In: *14th IEEE International Enterprise Distributed Object Computing Conference Workshops (EDOCW)*.

Buckl, S., C. M. Schweda, and F. Matthes (2010). "A design theory nexus for situational Enterprise Architecture management." In: *14th IEEE International Enterprise Distributed Object Computing Conference Workshops (EDOCW)*.

Buckl, S., F. Matthes, and C. M. Schweda (2009). "Classifying Enterprise Architecture Analysis Approaches." In: *Proceedings of the Second IFIP WG 5.8 International Workshop (IWEI)*.

Burgin, M. (2009). *Theory of Information: Fundamentality, Diversity and Unification*. World Scientific Publishing Company.

Caetano, A., C. Pereira, and P. Sousa (2011). "Generating multiple consistent views from business process models." In: *Lecture Notes in Business Information Processing, Research and Practical Issues of Enterprise Information Systems*. Springer-Verlag, Berlin, Heidelberg.

Caetano, A., A. Assis, and J. Tribolet (2012). "Using business transactions to analyse the consistency of business process models." In: *45th Hawaii International Conference on System Science (HICSS)*. IEEE, pp. 4277–4285.

Caetano, A., A. R. Silva, and J. Tribolet (2009). "A role-based enterprise architecture framework." In: *Proceedings of the ACM Symposium on Applied Computing*. ACM, pp. 253–258.

Davoudi, M. R. and K. Sheikhvand (2012). "An Approach towards Enterprise Architecture Analysis using AHP and Fuzzy AHP." *International Journal of Machine Learning and Computing* 2, 46–51.

Fettke, P. and P. Loos (2003). "Ontological Evaluation of Reference Models Using the Bunge-Wand-Weber Model." In: *Proceedings of the AMCIS 2003*.

Fischer, R., S. Aier, and R. Winter (2007). "A federated approach to enterprise architecture model maintenance." *Enterprise Modeling and Information Systems Architectures* 2, 14–22.

Franke, U., W. R. Flores, and P. Johnson (2009). "Enterprise Architecture Dependency Analysis using Fault Trees and Bayesian Networks." In: *Proceedings of the 2009 Spring Simulation Multiconference (SpringSim '09)*.

Genesereth, M. R. and N. J. Nilsson (1987). *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann, Los Altos, CA.

Gomez-Perez, A. and R. Benjamins (1999). "Overview of knowledge sharing and reuse components: Ontologies and problem-solving methods." In: *Proceedings of IJCAI-99 Workshop on Ontologies and Problem Solving Methods (KRR5)*.

Gruber, T. R. (1993). "A translation approach to portable ontology specifications." *Knowledge Acquisition* 5, 199–220.

Guarino, N., D. Oberle, and S. Staab (2009). "Handbook on Ontologies." In: Springer Berlin Heidelberg. Chap. What Is an Ontology? Pp. 1–17.

Hamalainen, N. (2008). "Evaluation and Measurement in Enterprise and Software Architecture Management." MA thesis. University of Jyvaskyla.

Iacob, M.-E., D. Quartel, and H. Jonkers (2012). "Capturing Business Strategy and Value in Enterprise Architecture to Support Portfolio Valuation." In: *16th IEEE International Enterprise Distributed Object Computing Conference (EDOC)*.

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

Johnson, P. and M. Ekstedt (2007). *Enterprise Architecture: Models and Analyses for Information Systems Decision Making*. Lightning Source Incorporated.

Johnson, P., R. Lagerstrom, P. Narman, and M. Simonsson (2007). "Enterprise Architecture Analysis with Extended Influence Diagrams." *Information Systems Frontiers* 9, 163–180.

Johnson, P., M. Ekstedt, E. Silva, and L. Plazaola (2004). "Using Enterprise Architecture for CIO Decision-Making:On the Importance of Theory." In: *Proceedings of the Second Annual Conference on Systems Engineering Research*.

Kurpjuweit, S. and R. Winter (2007). "Viewpoint-based Meta Model Engineering." In: *2nd International Workshop on Enterprise Modelling and Information Systems Architectures (EMISA)*.

Lagerstrom, R., P. Johnson, and D. Hook (2010). "Architecture analysis of enterprise systems modifiability - Models, analysis, and validation." *The Journal of Systems and Software* 83, 1387–1403.

Lankhorst, M. (2005). *Enterprise Architecture at Work: Modeling, Communication, and Analysis*. Springer.

Lemaignan, S. (2012). "Grounding the Interaction: Knowledge Management for Interactive Robots." PhD thesis. Universite de Toulouse.

Narman, P., P. Johnson, and L. Nordstrom (2007). "Enterprise Architecture: A Framework Supporting System Quality Analysis." In: *11th IEEE International nterprise Distributed Object Computing Conference (EDOC)*.

Niemann, K. D. (2005). *From Enterprise Architecture to IT Governance*. Friedr. Vieweg /& Sohn Verlag.

Olivé, A. (2007). *Conceptual Modeling of Information Systems*. Springer-Verlag, Berlin.

Peffers, K., T. Tuunanen, M. A. Rothenberger, and S. Chatterjee (2008). "A Design Science Research Methodology for Information Systems Research." *Journal of Management Information Systems* 24, 45–77.

Pinto, H. S., A. Goméz-Pérez, and J. P. Martins (1999). "Some issues on ontology integration." In: *Proceedings of IJCA99: Workshop on Ontologies and Problem Solving Methods*.

Saat, J., U. Franke, R. Lagerstrom, and M. Ekstedt (2010). "Enterprise Architecture meta models for it/business alignment situations." In: *14th IEEE International Enterprise Distributed Object Computing Conference (EDOC)*.

Sousa, P., A. Caetano, A. Vasconcelos, C. Pereira, and J. Tribolet (2006). "Enterprise Modelling and Computing with UML." In: ed. by P. Rittgen. Idea Group Inc. Chap. Enterprise architecture modeling with the UML 2.0, pp. 67–94.

Studer, R., R. Benjamins, and D. Fensel (1998). "Knowledge Engineering: Principles and Methods." *Data & Knowledge Engineering* 25, 161–198.

Tarr, P., H. Ossher, W. Harrison, and S. M. Sutton Jr. (1999). "N Degrees of Separation: Multi-dimensional Separation of Concerns." In: *Proceedings of the 21st International Conference on Software Engineering*. ICSE '99. Los Angeles, California, USA: ACM, pp. 107–119. ISBN: 1-58113-074-0. DOI: 10.1145/302405.302457.

The Open Group (2012). *ArchiMate 2.0 Specification*. Van Haren Publishing.

Uschold, M. and M. Gruninger (1996). "Ontologies: Principles, methods and applications." *Knowledge engineering review* 11, 93–136.

Vaculin, R. (2009). "Process Mediation Framework for Semantic Web Services." PhD thesis. Department of Theoretical Computer Science, Mathematical Logic, Faculty of Mathematics, and Physics, Charles University.

W3C (2012). *OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax*. W3C. URL: http://www.w3.org/TR/2012/REC-owl2-syntax-20121211/.

Wand, Y. and R. Weber (1993). "On the ontological expressiveness of information systems analysis and design grammars." *Journal of Information Systems* 3 (4), 217–237.

Yu, E. (1997). "Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering." In: *3rd IEEE Int. Symp. On Requirements Engineering (RE)*.

Zivkovic, S., H. Kuhn, and D. Karagiannis (2007). "Facilitate modeling using method integration: An approach uisng mapping and integration." In: *15th European Conference in Information Systems (ECIS)*.