MANAGING CONSISTENCY IN MULTI-VIEW ENTERPRISE MODELS: AN APPROACH BASED ON SEMANTIC QUERIES

Dimitris Karagiannis
University of Vienna, dk@dke.univie.ac.at

Robert Andrei Buchmann
Babeş-Bolyai University of Cluj-Napoca, robert.buchmann@econ.ubbcluj.ro

Dominik Bork
University of Vienna, dominik.bork@univie.ac.at

Follow this and additional works at: http://aisel.aisnet.org/ecis2016_rp

Recommended Citation
Karagiannis, Dimitris; Buchmann, Robert Andrei; and Bork, Dominik, "MANAGING CONSISTENCY IN MULTI-VIEW ENTERPRISE MODELS: AN APPROACH BASED ON SEMANTIC QUERIES" (2016). Research Papers. 53.
http://aisel.aisnet.org/ecis2016_rp/53

This material is brought to you by the ECIS 2016 Proceedings at AIS Electronic Library (AISeL). It has been accepted for inclusion in Research Papers by an authorized administrator of AIS Electronic Library (AISeL). For more information, please contact elibrary@aisnet.org.
MANAGING CONSISTENCY IN
MULTI-VIEW ENTERPRISE MODELS:
AN APPROACH BASED ON SEMANTIC QUERIES

Research

Karagiannis, Dimitris, University of Vienna, Vienna, Austria, dk@dke.univie.ac.at
Buchmann, Robert Andrei, Babeş-Bolyai University of Cluj-Napoca, Romania, robert.buchmann@econ.ubbcluj.ro
Bork, Dominik, University of Vienna, Vienna, Austria, dominik.bork@univie.ac.at

Abstract

Enterprise modelling is one of the popular means for capturing organizational knowledge in representations that are both human-readable (that is, diagrammatic) and machine-readable (that is, sufficiently formal and granular). However, the discipline faces certain challenges pertaining to the complexity of the socio-technical system to be modelled. This often requires a separation of concerns, to allow separate modelling of various enterprise facets – work processes, organizational structure, resource descriptions. This separation must be compensated by consistency management approaches, as diagrams of different types become inter-dependent views, enabled by different viewpoints (modelling language fragments with different scopes). The paper introduces novel means for managing the consistency of knowledge captured in multi-view enterprise models. The proposal is based on semantic graphs derived from diagrammatic models, and queries acting upon these. To achieve the goal, the Linked Data paradigm is repurposed and its technological enablers are aligned to the underlying graph nature of enterprise models. Several use cases will be discussed: (a) view transformations through graph rewriting; (b) view synchronization through reasoning; (c) passive view consistency checks. Exemplary cases are extracted from two research projects where the proposal has been successfully applied, therefore background on the projects will be provided to facilitate understanding.

Keywords: Model queries, Multi-view Enterprise Modelling, Linked Data, View consistency.

1 Introduction

Knowledge management systems (KMS) rely on repositories that employ a variety of knowledge representation techniques, depending on requirements and intended usage. The tradition of knowledge management (in the sense defined by (Jennex 2005) (Nakamori 2011)) typically employs human-oriented representations captured in different types of content (handbooks, documented work procedures, diagrams, hypertext) exposed and shared through knowledge sharing channels (e.g., collaborative content management systems, wikis, enterprise portals etc.). On the other hand, the tradition of knowledge engineering (as evolved from the discipline of Artificial Intelligence) places emphasis on machine-interpretable knowledge representations that can be retrieved, filtered or reasoned upon with automated means and various degrees of granularity or expressivity (i.e., knowledge bases built on semantic networks, conceptual graphs, description logics). A comprehensive survey of this rich variety is available in (Maier, 2007). One of the approaches covered by the survey is that of enterprise modelling as a means of knowledge externalization, also recognizing the need for multi-perspective enterprise modelling (Frank, 2002) (Scheer, 1992) that employs visual modelling languages in order to describe the various facets of an enterprise (e.g., processes, systems, resources). This enables the accumulation of a model-based organizational memory/knowledge base, while also contributing to the success factors of KMS as identified in (Jennex et al., 2008) (e.g, "enterprise-wide
knowledge structure that is clearly articulated", "a knowledge strategy that identifies users, sources, processes…"). In addition to providing visual representations designed for human communication, sense-making and knowledge sharing, enterprise modelling may also employ formal enablers (metamodels, metamodelling platforms, formal language grammars) to enable machine-interpretability and automation of certain reasoning tasks such as those involved in consistency management across different enterprise facets – the key challenge to be addressed in this paper.

An inherent requirement for managing the complexity of enterprise models emerges from the multiple perspectives that must be supported - either with multiple modelling languages, or with a single language that enables multi-view modelling – the latter implying that different types of models support different perspectives (here to be considered "viewpoints"). Consequently, view semantics must be leveraged in order to ensure consistency across these perspectives. This cannot be achieved with diagrammatic representations that lack machine-interpretable semantics (e.g., diagrams based on Powerpoint shapes). Instead, fully-fledged modelling methods must be designed and implemented to bridge human-oriented diagrammatic knowledge representations and machine-readable representations, thus benefitting from advances in knowledge engineering in order to improve the value of externalized knowledge for both human modellers and model-driven information systems.

The goal of the paper is to introduce a semantic query-based approach for preserving the consistency of knowledge captured in multi-view enterprise models. More specifically, the work at hand outsources consistency preservation techniques to a Linked Data environment where model structure and semantics are exported as semantic networks for graph processing. To this aim, we repurpose the practices and technological space of the Linked Data paradigm (as outlined by (Heath and Bizer, 2011)) for the underlying graph nature of multi-view enterprise models. The proposal will be illustrated by several view consistency preservation use cases: (i) view transformation; (ii) view synchronization; (iii) passive view consistency checks. Modelling method engineers and tool developers may consider the proposed approach in relation to enterprise modelling requirements, since the proposal is reusable outside the context of the use cases discussed here, and for a wide class of modelling languages.

The remainder of the paper is structured as follows: Section 2 positions the research challenge to be addressed and discusses several related works. Section 3 describes two case studies derived from two research projects for which the authors developed multi-view enterprise modelling methods and tools: SOM (Ferstl and Sinz, 2005) and ComVantage (Karagiannis et al., 2014). These are aimed at establishing context for the use cases where the view consistency preservation techniques are applied in several examples discussed throughout Section 4. The paper ends with a concluding SWOT evaluation that suggests opportunities for further developments.

2 Problem Statement and Background

2.1 Challenges in multi-view enterprise modelling

The paradigm of enterprise modelling builds on the tradition of business process modelling by integrating procedural knowledge (expressed in business process models), organizational knowledge and various aspects of the enterprise context (capabilities, resource descriptions etc.) that are relevant for the organizational memory or with respect to knowledge-driven information systems. A global requirement emerges for decomposing the overarching ontological description of an enterprise in multiple types of models, each expressing a different viewpoint on the same complex system. Popular enterprise frameworks show various ways in which this decomposition is achieved across different layers/facets/scopes - see the rather ontological approach of the Zachman framework (Zachman, 1987), the ARIS perspectives (Scheer, 1992) or the modelling methods promoted by The Open Group, i.e. ArchiMate (OpenGroup, 2015a), TOGAF (OpenGroup, 2015b). Knowledge management
frameworks recognize quite similar building blocks – see (Pawlowski and Bick, 2012), suggesting a convergence between the paradigms of knowledge management and enterprise modelling.

The semantics of this decomposition may vary from generic (e.g., part-of relations: business process models decomposed in automated workflows) to explicit (e.g., a product model "motivates" a production process model). Multi-view modelling is required when different facets are constrained by dependencies across their different viewpoints (typically this means that changes in a model should propagate throughout models of other types). A viewpoint, in this context, represents a metamodel partition that isolates concepts/properties pertaining to some specific modelling scope/requirement. Dependencies across views may manifest in different ways:

a) different viewpoints may share common concepts (although this is a case of redundancy, it may be helpful for sense-making and comprehensibility to analyse different sets of properties for the same entity, in different types of models);

b) relations of various semantic richness may be established between concepts that belong to different viewpoints (e.g., possibly manifested as hyperlinks between related models);

c) computable dependencies may constrain elements of different models.

The goal of the work at hand is to capture the intra-view and inter-view semantics in a uniform, abstract graph-based representation, so that different views on the same enterprise can be kept semantically consistent with a model query-based approach. We also extend this goal with the requirement that views must be made available to a Linked Enterprise Data environment (as defined by (Wood, 2010)), where not only human modellers, but also run-time systems may rely on view consistency. Therefore we formulate the research question as follows: How can one leverage the enterprise model semantics in order to preserve consistency across different views, in an application environment based on Linked Data?

Means for achieving the goal vary and will be discussed here for several exemplary instance cases: (i) consistent views can be automatically produced by transforming other views; (ii) editing operations on a view can be automatically synchronized in related views; (iii) passive consistency checks can signal possible deviations between views (then the deviations are manually corrected).

2.2 Related works

Model queries have been employed in the existing literature for a variety of purposes. Recent design works that propose generic model query languages driven by a meta-metamodel (rather than a specific model type) are available in the form of GMQL (Delfmann et al., 2015) or VMQL (Störrle, 2011) – however their scope is limited to retrieval queries through visual means, strictly within a modelling tool (as a modelling plug-in functionality). They do not consider inter-view dependencies and model rewriting based on those dependencies, nor do they address the requirement of exposing multi-view semantics to external knowledge-driven systems. Other model query approaches have their scope limited to business process models for process-aware information systems (Beeri et al., 2008) (Wang et al., 2013) (Sakr and Awad, 2010), hence they rely on the semantic specificity of BPM standards (e.g., BPMN) and, again, are not concerned with multi-perspective modelling or with exposing multi-view models to knowledge systems outside the modelling tool. The work at hand aims to be reusable for arbitrary types of models enabled by a metamodelling platform, although business process models have also been involved in our use cases, as procedural views on an enterprise description. XML vocabularies - e.g., BPEL (OASIS, 2015), XPDL (WfMC, 2015) - are typically preferred for repositories of model serializations, since they have been used for syntactic interoperability goals, before the advent of the Linked Data paradigm. The proposal of this paper extends towards semantic interoperability and enriches the concept of Linked Open Data with that of "Linked Open Models" where semantic queries enable specific ways of processing a graph-based representation of model structure and semantics.
Multi-view modelling has been traditionally employed to facilitate separation of concerns in software engineering, in standards such as ISO/IEC/IEEE 42010:2011 (ISO/IEC/IEEE, 2015), ANSI/X3/SPARC (Brodie and Schmidt, 1982) or the 4+1 model (Kruchten, 1995). A taxonomy of view-based modelling approaches was provided by (Goldschmidt et al., 2012) and a set of inter-viewpoint formal generic rules is provided in (Nuseibeh et al., 1994). A further conceptual classification of relations between views can be found in (Boucke et al., 2008), along the dimensions of usage, scope and mechanism. The work of (Masuda et al., 2014) provides a multi-view method for services without considering consistency aspects. Compared to this baseline, the paper at hand proposes the management of view consistency by applying query-based reasoning techniques on semantic graphs derived from the underlying graph nature of enterprise models. The focus of multi-view modelling is thus shifted from its originating context (software architecture) to enterprise modelling, where organizational knowledge is typically fragmented across multiple viewpoints.

2.3 Conceptual enablers

We propose a possible instantiation of the theoretical framework of (Bork et al., 2015), where multi-view consistency preservation techniques have been formally defined and compared. However, the paper at hand aims at being self-contained and pragmatic in its proposal, therefore it will not make use of the formal notations of the theoretical framework. Instead, we assume reader’s familiarity with the knowledge representation techniques surveyed in the KMS monograph (Maier, 2007), particularly those pertaining to conceptual modelling and semantic integration. The proposal exploits semantic integration techniques by outsourcing consistency preservation tasks to a Linked Data environment. An alternative instantiation, based strictly on metamodelling platforms, was provided in (Bork, 2015).

The paper aligns to the notion of a modelling method as introduced by (Karagiannis and Kühn, 2002) and specialized as a knowledge-based enabler for business process management in (Karagiannis and Woitsch, 2015). According to the definition, a modelling method comprises several building blocks: (i) a modelling language that establishes the modelling constructs (concepts and relations integrated in a metamodel) and their semantics (explicit properties in concept schemata); (ii) a modelling procedure that establishes steps to be followed in order to achieve various modelling goals; (iii) mechanisms and algorithms that process the knowledge captured in diagrammatic models. Table 1 summarizes requirements that these building blocks must satisfy in order to enable the proposal of this paper.

<table>
<thead>
<tr>
<th>Requirements for multi-view modelling</th>
<th>Requirements for leveraging view semantics</th>
</tr>
</thead>
</table>

**Table 1. Requirements on the modelling method building blocks**

3 Case Studies and Requirements

A corpus of enterprise modelling methods and their tool implementations accumulated in the Open Model Initiative Laboratory (OMiLAB, 2015b) served as a starting point for investigating the partitioned nature of enterprise modelling languages in relation to complexity management requirements. Furthermore, the authors participated in the conceptualization and implementation of
two project-based modelling methods from which illustrative examples are presented in Section 4. The current section describes these project-based cases and their relevant requirements.

3.1 The ComVantage project case

The ComVantage method enables the modelling of a "ComVantage enterprise" (i.e., virtual enterprise whose operations require IT support in the form of mobile apps capable of consuming shared Linked Data repositories). A ComVantage enterprise can be modelled on multiple layers of abstraction (scopes), across two types of facets – behavioural and structural, as expressed in Table 2. The facets and the scopes provide the aforementioned partitioning of the overarching enterprise metamodel across different viewpoints expressed in different types of models with various dependencies - some of them domain-specific, others relatable to the more generic frameworks mentioned in Section 2.1. Details on the metamodel are available in public deliverables (Karagiannis et al., 2014), we only provide in Table 2 a brief explanation about what each type of model can describe. We will further isolate a fragment of the method (from the Requirements and App execution scopes) to illustrate the proposed approach.

<table>
<thead>
<tr>
<th>FACETS:</th>
<th>Behavioural</th>
<th>Structural</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Procedural Views:</strong> (procedural knowledge captured in the form of flowcharts with varying semantics and customized notation)</td>
<td>Models that describe value exchanges between entities that participate in the business model or in an enterprise-level process</td>
<td>Models that describe the values that are created by the business or by each process in particular (e.g., product features, product-service mixtures, abstract values) in terms of a flavour of &quot;feature modelling&quot; approach (Kang et al., 1990)</td>
</tr>
<tr>
<td><strong>Collaborative Views:</strong> (the same kind of knowledge, expressed as interactions in order to highlight necessary interfaces)</td>
<td>Models that describe how different resources must interact based on their mappings on work processes</td>
<td>Models that describe required and available resources (mobile apps, data endpoints, skills of human resources, knowledge items)</td>
</tr>
<tr>
<td><strong>Motivator Views:</strong> (structural descriptions of the values that must be created by the enterprise)</td>
<td>Models that describe how mobile apps must interact according to the flow of the process they must support</td>
<td>Models that describe mobile apps that are required and must be &quot;orchestrated&quot; (Ziegler et al., 2012) to support a process</td>
</tr>
<tr>
<td><strong>Participants Views:</strong> (structural descriptions of available or required resources, liable entities and their capabilities)</td>
<td>Models that describe mobile apps that are required and must be &quot;orchestrated&quot; (Ziegler et al., 2012) to support a process</td>
<td></td>
</tr>
</tbody>
</table>

Table 2. The ComVantage model type stack overview

Fig. 1 describes a business process model with activities mapped (through notational hyperlink anchors) on their required IT support (mobile app features). It also shows a different view on the same business process, this time expressed in terms of the usage precedence for the required mobile apps. Beyond the need to keep these views consistent for the modeller, this procedural knowledge also becomes input for an app orchestration engine (Ziegler et al., 2012) which automatically chains and executes apps in the order dictated by the process flow; in another scenario, the same model may be
employed as a transfer-of-training facilitator (in the sense defined by (Amrou et al. 2015)), to train employees in acquiring knowledge on new work procedures and how apps support them.

Figure 1. Process view and app-focused view on the same business process (created with the tool available at (OMiLAB, 2015a))

The dependencies between these views are intricate and cannot be enforced by the default metamodeling constraints (i.e., domain, range or cardinality checks for visual connectors). For example, all apps from the app-focused view must be linked to some activities in the process view and the chained app order (in the app-focused) view must be determined by the activity flow in the process view. Sections 4.1 and 4.3 will illustrate how these dependencies are handled by the proposed approach.

3.2 The Semantic Object Model (SOM) case

The Semantic Object Model (SOM) method, developed by (Ferstl and Sinz, 2005) also enables the analysis enterprises using a multi-view approach (see its perspectives in Fig. 2), however in less domain-specific terms. First, the enterprise representation is decomposed into three layers: an Enterprise Plan layer taking an outside perspective on the enterprise; a Business Process and a Resource layer from an internal perspective. A SOM Business Process model is established by four interrelated views enabled by different schemata (the “viewpoints”):

- **Interaction Schema** (IAS) provides a structural view on the process by investigating the business objects (i.e., environmental objects and objects of discourse), that are coordinated by business transactions (i.e., enforcing transactions, contracting transactions, initiating transaction, control transaction, and report transaction);
- **Task-Event Schema** (TES) focuses on the behavioural aspects, with tasks connected by events and triggered by business transactions;
- **Object Decomposition Schema** (ODS) visualizes the hierarchical decomposition of business objects in a tree-based structure;
- **Transaction Decomposition Schema** (TDS) visualizes the hierarchical decomposition of business transactions in a tree-based structure.
Fig. 3 illustrates a sample SOM Business Process model split across the four views: TDS on the upper left; ODS on the lower left; IAS on the upper right; and TES on the lower right corner. Two business objects are considered in the process (visualized in the ODS): a (object of discourse) seller, and a (environmental object) buyer coordinated initially by the enforcing transaction "sell products". This transaction, according to the TDS, has been decomposed into an initiating transaction "advertise products", a contracting transaction "negotiate conditions", and an enforcing transaction "sell product". The structural view is depicted by the IAS view - the coordination of the two business objects by means of the three business transactions, i.e., in this case specifying a communication pattern.

Finally, the behaviour is depicted by the TES as a sequence of tasks executed in order to realise the process. Every business transaction is represented by a corresponding send and receive task in the TES. These tasks are put in sequence by means of events: first the advertise products tasks are executed, followed by the negotiate conditions tasks, and concluded by the sell product tasks. The example illustrates the rich semantic dependencies, e.g., business objects and business transactions of the IAS must be defined previously in the ODS and TDS, respectively. This all the more comes to notice when considering the modelling procedure of SOM. SOM comes with a set of predefined decomposition rules that modellers must apply in order to refine an initial SOM business process model. Applying these decomposition rules in one view inevitably causes inconsistencies in some other views, unless a synchronization mechanism is in place, such as the one proposed in Section 4.2.

Figure 2. The frame of the SOM enterprise architecture (OMiLAB, 2015c)

Figure 3. Multi-view SOM model, created with the tool available at (OMiLAB, 2015c)
4 Semantic Queries on Multi-view Enterprise Models

4.1 The underlying schema for anchoring model queries

The Linked Data paradigm (Heath and Bizer, 2011) and its technological space, based on the graph data model RDF (W3C, 2015c), is repurposed here to capture the underlying graph nature of multi-view enterprise models in a uniform representation across multiple views. A mechanism is hereby employed to serialize views as semantic graphs that can be reasoned upon with various means. This paper emulates reasoning through SPARQL queries (W3C, 2015c). The input for the serialization mechanism is the multi-view metamodel together with the view contents exported from a modelling tool. The implementation of such a mechanism resulted from the authors' work in the ComVantage project (OMiLAB, 2015a) and is reusable for any modelling tool built on the ADOxx metamodeling platform (available for open use at (BOC, 2015)). A formal description of the produced knowledge structure has been provided as a hypergraph formalism in (Karagiannis and Buchmann, 2016), therefore only the key design decisions are summarized here, to ensure that the text is self-contained.

The model serialization relies on a 3-layered RDF Schema (see Fig. 4) derived from the abstraction layers that are commonly recurring in popular metamodelling platforms such as (BOC, 2015) (Kelly et al., 2013).

Figure 4. The RDF Schema layers for model serialization

The meta\textsuperscript{2} layer maps abstract constructs of common meta\textsuperscript{2}models (Kern et al., 2011) to primitive RDF Schema concepts (e.g., rdfs:class, rdf:property) and derives modelling-relevant specializations for them (e.g., the class of all views, all modelling concepts, all connectors, all inter-view hyperlinks, all properties of view elements). The meta layer generates from the language elements (i.e., metamodel) specializations of the above mentioned primitives (e.g., the class of all Apps, Tasks, Actors, or other language-specific concepts). The model layer generates RDF graph patterns from diagram patterns detected in the views, and links them to the abstractions on the upper layers. All views are considered attributed directed labelled graphs (same as in the model query approach of (Delfmann et al., 2015)) but are integrated in a common hypergraph through semantic links derived from inter-view hyperlinks and other referencing properties found in model elements. The basic principles for this derivation are, as follows (with respect to the meta\textsuperscript{2} layer labels in Fig. 4):

Each view is serialized as a “named graph” (instance of cv:Model), annotated with model metadata; each editable property of a model element is serialized as an RDF predicate (instance of cv:EditableProperty), with the exception of tabular properties which become rdf:List structures; each
inter-view hyperlink is serialized as an RDF predicate (instance of cv:NonattrRelation); each visual connector with no editable attributes is serialized as an RDF predicate (instance of cv:NonattrRelation); each visual connector which has its own attributes is serialized as an n-ary RDF relation, a pattern explained in (W3C, 2015a). Finally, some helper relations are prescribed by the schema for modelling-specific cases (e.g., to serialize visual containments, to reference the target of an inter-view hyperlink outside its originating view).

Since URIs must identify all model elements in a Linked Data environment, they are generated from a namespace provided by the modeller (usability mechanisms such as generation of model elements with pre-filled URIs are possible in the tool implementation). Arbitrary RDF statements may be included by the modeller, beyond the semantics prescribed by the modelling language semantics. Model deserialization is also enabled by preserving the canvas position of model elements as exported predicates, making it possible to import back in the modelling tool the results of graph processing, such as those to be discussed in the following sections.

4.2 Semantic queries as view transformation rules

Reasoning on RDF graphs is possible through a variety of means, the most popular being OWL ontologies (W3C, 2015b), SWRL rules (W3C, 2015d), but also SPARQL queries (W3C, 2015c) that emulate production rules to infer new relations in the graph (CONSTRUCT or a mix of INSERT/DELETE), therefore graph rewriting can be applied by executing such relation-generating queries. Once the aforementioned model examples are exported using the RDF serialization schema discussed in Section 4.1, they act as semantic networks and the views (both intra-view connectors and inter-view hyperlinks) can be traversed by semantic queries. In Fig. 1 the dependencies between the two views may be expressed in the form of an inference: IF a user must perform an activity ActX after they perform an activity ActY that requires AppY THEN AppX must be used (chained) after AppY. The orchestration diagram ("app-focused view") is nothing else than the conclusion of this inference, applied on the procedural knowledge expressed in the business process model. This reasoning may be implemented as a set of graph rewriting rules, executed in the sequence expressed by Fig. 5 (labels in the diagram are not meant to be readable, only the structural changes in the graph). Table 3 expresses the same rules as a sequence of SPARQL updates which will transform the input graph (top-left diagram in Fig. 5) into the output graph (bottom-right diagram).

| Step5. | DELETE {?x ?y ?pe1.} WHERE { {?x ?y ?pe1. ?pe1 rdf:type cv:PE.}} |

Table 3. Examples of view transformation rules in SPARQL (based on the graph rewriting rules discussed in (Karagiannis and Buchmann, 2016))
As a result of the execution, RDF statements expressing app precedence (AppX must be used after AppY) are added to the semantic graph repository, as a serialization of the app-focused model in Fig. 1. How the output is handled depends on the intended usage of the view contents. On one hand, they can be deserialized back in the same modelling tool, based on the same RDF schema, plus positioning information which is also exported as an RDF property of each element. However, a more relevant use, already suggested in the problem statement and research question (Section 2.1) is that the knowledge expressed in multi-view models should be exposed to a Linked Enterprise Data environment where knowledge-based systems may benefit from it, as it is the case of the overall ComVantage project. An example of such a system is the app orchestration engine which deploys runtime apps that are chained according to the diagram information (Ziegler et al., 2012). Another usage is suggested here, in Fig. 6:

On the left side, the app orchestration model is extended with additional semantic links describing an app in terms of (i) its interaction elements (abstract UI elements) linked to (ii) interaction steps from a...
control flow representing expected UI interactions. These extensions are exported to the Linked Data repository using the same serializer mechanism.

On the right side, a "knowledge stepper" training tool emulates the mobile app UI and highlights its interaction points while advancing step by step through a representation of the interaction workflow, outside the modelling tool (based on querying the "app view" model contents). This can be used on one hand to validate app requirements in relation to the supported business process (if we work with "to-be" models) and, on the other hand, as a knowledge sharing tool to train employees in how the mobile app orchestration supports their work procedures (if we work with "as-is" models). Consistency between the training contents and the design-time business process is ensured since the training tool advances through step-by-step model queries along the visual connectors visible in the models.

![Diagram showing a knowledge-driven system](image)

**Figure 6.** View-awareness induced to a knowledge-driven system outside the modelling tool

### 4.3 Semantic queries as view synchronization rules

For the synchronization case we select an example from the SOM method, whose modelling tool includes synchronization mechanisms to propagate changes from one view to the other views (see the discussion on Fig. 3).

This task may also be delegated to query-time production rules applied on the model serialization in a Linked Data environment. The graph derived from one view (e.g., TDS, IAS, ODS) is extended based on patterns detected in the graph corresponding to another view, based on some synchronization rules. Fig. 7 shows, on the left side, pairs of views to be synchronized and on the right side the production queries that achieve the synchronization in the abstract graph.

An alternative strategy is to replace queries with custom rules that will produce the new statements on the fly, every time the content of the graph is edited. For this to work, the RDF repository acting as a model repository must support reasoning based on custom inference rules, if the graph repository allows it (e.g., GraphDB provides its own rule syntax for this (OntoText, 2015), Sesame recently introduced support for SPIN and persistent custom inference rules (OpenRDF, 2015)).
4.4 Semantic queries for passive view consistency checks

Finally, reasoning-driven view editing may be complemented by rather passive consistency checks which only signal deviations without affecting existing contents. Relative to the ComVantage model samples discussed in Section 3.2, we formulate in Table 4 some brief examples of such consistency checking scenarios, together with the SPARQL queries that can perform these checks (namespaces will be omitted for readability).

Similar checks may be devised to detect deviations in the SOM models (e.g., *are all business objects/business transaction in the IAS view previously introduced in the ODS/TDS?*)

<table>
<thead>
<tr>
<th>Check: are all the apps involved in an orchestration also linked to the business process model (to check deviations)?</th>
<th>Check: if (in the orchestration) one app must be used after another app, is it true that in the business process model the activities they support are in the same order?</th>
<th>Check: is there an app-supported activity in the business process model for which there is no corresponding app present in the app orchestration?</th>
</tr>
</thead>
</table>

Table 4. Passive view consistency checks expressed as SPARQL queries

5 Concluding SWOT Analysis

The work at hand complements the theoretical proposal in (Bork et al., 2015) with a pragmatic deployment based on a model serialization mechanism and the processing of enterprise views as semantic graphs. Therefore the paper stimulates interdisciplinarity in two dimensions: (i) a
convergence of techniques originating in the paradigms of conceptual modelling and Semantic Web, employed here to leverage view semantics; (ii) the application of knowledge engineering techniques in support of knowledge management concerns – that is, in support of managing the inherent complexity of knowledge externalized as multi-view enterprise models. Modelling method engineers may integrate such mechanisms in knowledge management systems that rely on diagrammatic representations and enterprise modelling tools.

To illustrate the proposal, several project-based use cases were discussed in the paper: view transformation, view synchronization and consistency checking. The feasibility of the approach has been validated in the (OMiLAB, 2015b) research environment by implementing enterprise modelling tools for exemplary cases that raise requirements for multi-view modelling. The project experience and the discussed examples are the basis for the following SWOT evaluation:

**Strengths**: The proposal enables multi-view enterprise models that capture organizational knowledge in diagrammatic form to be exposed as semantic graphs in order to tackle consistency challenges pertaining to multi-perspective modelling and “view-aware” knowledge systems.

**Weaknesses**: Usability has been of low priority to the existing project goals, meaning that model queries have been incorporated in external systems or typed manually in some SPARQL user interface. User-friendly interfaces for handling semantic queries over enterprise models are still needed to facilitate this.

**Opportunities**: Meta-modelling platforms may benefit on the long term from employing an RDF-based model storage instead of the traditional relational databases, further opening opportunities for linking and sharing model semantics across views created by different organizations in the same virtual enterprise, within a Linked Enterprise Model cloud.

**Threats**: The general uptake of the Linked Data paradigm is quite slow, mostly due to the difficulties of explaining to business stakeholders the benefits of distributed semantic graphs against traditional data repositories. However, we consider that the hereby presented approach opens opportunities for revealing benefits not previously grasped, nor advocated with respect to Linked Data adoption.
References


