Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 00 (2017) 000–000

Procedia Computer Science

www.elsevier.com/locate/procedia

21st International Conference on Knowledge Based and Intelligent Information and Engineering Systems, KES2017, 6-8 September 2017, Marseille, France

Multi-Context Systems for Consistency Validation and Querying of Business Process Models

Nikolaos Lagos[a], Adrian Mos[a], Jean-Yves Vion-Dury[a] *

[a]*Xerox Research Centre Europe, 6 chemin de Maupertuis, Meylan 38240, France*

Abstract

Large organizations today face a growing challenge of managing heterogeneous process collections containing business processes. Explicit semantics inherent to domain-specific models can help alleviate some of the management challenges. Starting with concept definitions, designers can create domain specific processes and eventually generate industry-standard BPMN for use in BPMS solutions. However, in such a multi-layered setting, any of these artefacts (concepts, domain processes and BPMN) can be modified by various stakeholders and changes done by one person may influence models used by others. There is therefore a need for tool support to aid in keeping track of changes done and their impacts on different stakeholders. In this paper, we present a multi-context systems based approach that allows inferring impacts of changes, especially in terms of consistency, and executing semantic queries. In contrast to existing work, our framework allows the co-existence of different formalisms, with potentially different characteristics, offering greater flexibility in knowledge base and tool integration.

Peer-review under responsibility of KES International.

Keywords: consistent process management; domain-specific process model; multi-context system

1. Introduction

Large organizations today face a growing challenge of managing heterogeneous process collections containing business processes that correspond to various practices and domains. Explicit semantics inherent to domain-specific models can help alleviate some of the management challenges[1]. Starting with concept definitions, designers can

* Corresponding author.
E-mail address: nikolaos.lagos@xrce.xerox.com

Peer-review under responsibility of KES International.

create domain specific processes and eventually generate industry-standard Business Process Modelling Notation (BPMN)[2] for use in BPM Software (BPMS) solutions. The domain-specific and BPMN layers may be connected via an intermediate layer, responsible for taking domain specific modelling artifacts (process files, organizational structures) from a variety of tools in their respective form, and generating artifacts used by other tools, for instance BPMN enabled editors[3]. We refer to this approach as Multi-layered Business Process Modelling (MBPM).

Governance in MBPM is difficult as different tools and formalisms can be used for modelling. In addition, changes in one layer may influence models used in the other layers. Whether changes propagate to other layers (and how much) is determined by the transformation semantics between layers. However, today when looking at a large collection of processes it is almost impossible for various stakeholders to understand problems with process evolutions, changes, and inconsistencies with business goals. There is a need for tool support to detect inconsistencies, due to model updates happening in different layers, and address stakeholders' questions based on their own perspectives (business vs. architect vs. technical). In that respect, in prior work we proposed an ontology-based representation of the modelled artefacts[4]. However, we did not consider using specialized formalisms and reasoning engines tailored to different layers, neither how changes happening in one layer may influence the rest.

In this paper, we extend our previous work[4] by: (i) formally defining MBPM systems in terms of the knowledge represented in the various business modelling layers; (ii) describing how business modelling artefacts found in each layer can be transformed into semantic-based representations, taking into account the specificities of each layer; (iii) framing the governance of the semantic representations, and in extension that of the business artefacts, in a single integrated framework based on the multi-context systems (MCS) framework. MCS supports the co-existence of different formalisms, offering advanced inconsistency detection and querying capabilities for improved governance. With this work, we extend the field of semantic-based process modelling for multi-layered BPM systems.

In Section 2 we illustrate the challenges tackled in this work. Section 3 includes a short overview of the basic notions of MCSs. In Section 4 we define how MBPM layers can be transformed to MCS contexts. Section 5 presents how we resolve our challenges accordingly. Related work and conclusions are presented in Sections 5 and 6.

2. Problem Description

MBPM systems include three different types of layers: a Domain-Specific Process Modelling Language (DSPML) layer where a domain expert can design and model business processes and related information (e.g. concepts) using language(s) of his/her choice; an Inter-Layer Connection Bridge (ILCB) layer that transforms the information represented in the DSPML to models expressed in BPMN and vice-versa; and the BPMN (BPMN) layer that includes the generated BPMN models[1,3]. In that context, we have to cope with the following challenges.

Challenge 1: Keep Information Consistent Across Layers. Changes done in one layer may generate inconsistencies in the models of the other layers. Consider the example domain-specific definition in Fig. 1(a) (graphical and textual DSPML layers) with a corresponding sample process definition in Fig. 1(b), as it would be rendered in a domain-specific studio for the domain. The model describes a simplified view of an administrative workflow that takes place once a patient arrives at a hospital counter. The corresponding generated BPMN model is illustrated in Fig. 2(a). The BPMN model will be manipulated and modified by an expert in BPMN rather than a domain expert. The BPMN expert will naturally have different optimization aspects in mind. A typical case for instance could be that the subprocess related to the fourth task in the domain model is modified into a Service Task in the BPMN layer (Fig. 2(b)). In this case, an inconsistency should be detected, as the business requirement defined in the domain-specific concept specifies that it MUST be a subprocess.

Challenge 2: Support Querying for Different Layers. A typical need when governing a large number of business processes is executing appropriate queries to check whether existing Service Level Agreements (SLAs) are respected. For instance, in our case an SLA defines whether a patient will be transferred with an ambulance to a specialized viral infection clinic or not, and the time that a patient has to wait before entering quarantine according to the diagnosed disease (Fig. 1). To check if the SLA is respected we have to be able to respond to the following.

- What is a viral infectious disease e.g. is measles one? (Question 1)
- What does it mean more than 30-minute duration? (Question 2)

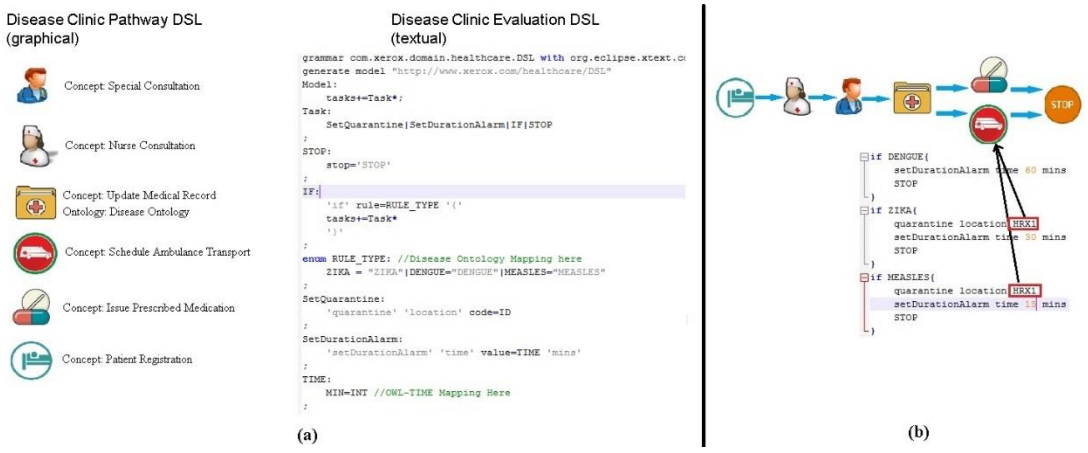

Fig. 1. (a). Domain specific concepts and (b). the domain specific model using them.

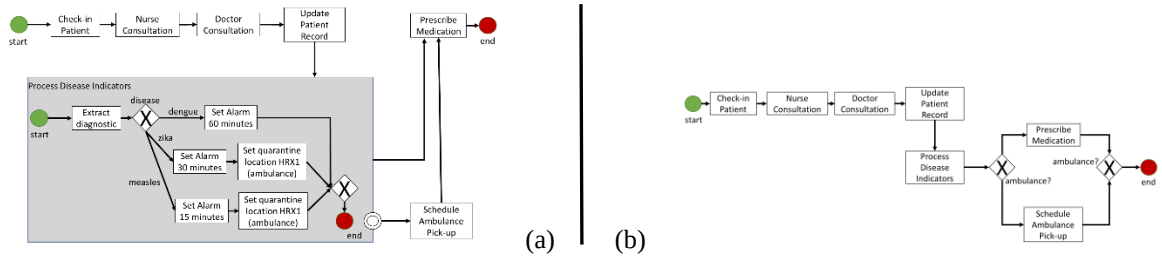

Fig. 2. (a). Possible initial BPMN model originating from the domain-specific one. Process Disease Indicators is represented as a Subprocess. (b). Modified BPMN model. Process Disease Indicators is represented as a Service Task.

Background knowledge available in third-party knowledge bases can help replying to the above questions. For instance, whether measles is a viral infectious disease or not, can be found in medical ontologies such as SNOMED-CT[5] and the Disease Ontology (DO)[6]. On the other hand, the temporal inference required in Question 2 can be supported by specialized temporal inference engines and ontologies such as OWL-Time[7]. However, the above knowledge bases are expressed in different formalisms and corresponding tools will be based on different reasoning profiles. We introduce a method that enables heterogeneous knowledge bases to co-exist in the same framework.

3. Background: Multi-Context Systems

The Multi-Context Systems (MCS)[8] theory has received increasing attention in recent years, especially in the knowledge representation community. MCS frames how knowledge bases in different formalisms can be interlinked.

The theory is based on the notion of *context*: a combination of a *knowledge base*, expressed in a specific *logic*, and a set of *bridge rules* that control the information flow between contexts. As described in Brewka et al.[9], a logic L is a 3-tuple $< KB_L, BS_L, ACC_L >$ over a signature Σ_L where KB_L is the set of well-formed admissible knowledge bases of L, BS_L is the set of possible belief sets, and $ACC_L : KB_L \rightarrow 2^{BS_L}$ is a function describing the semantics of the logic L. Bridge rules can be of the form $(c_1 : h) \leftarrow (c_2 : p_1), \ldots, (c_j : p_j), not\,(c_{j+1} : p_{j+1}), \ldots, not\,(c_m : p_m)$, where $1 \leq c_i \leq n$ (where n is the number of contexts), p is an element of some belief set of L, and c_1 refers to the context receiving formula h.

The notion of consistency in MCS is mainly related to the existence of an equilibrium: an acceptable belief state where each context takes the heads of all bridge rules that are applicable in the state into account[10]. But other notions of consistency can also be defined e.g. a restricted set of belief states allowed among all possible belief states. Querying across contexts is an active topic of research[11].

MCSs are especially interesting in our case as they allow: the co-existence of different formalisms in the same framework; rich inter-contextual information exchange (i.e. with the help of bridge rules); and consistency detection in heterogeneous and potentially decentralized settings.

4. Mapping Multi-Layer BPMs to Multi-Context Systems

To tackle the challenges described in Section 2, we propose generating MCS contexts, starting from the BPM layers. We expect that appropriate mappings will express the transformations to take place, as illustrated in Fig. 3.

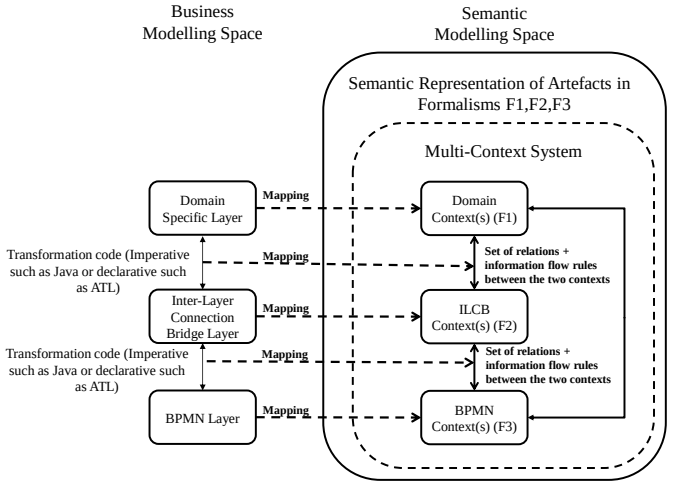

Fig. 3. Overview of our framework

In the remaining of this section, we define formally the represented objects and related knowledge found in the BPM layers (Business Modelling Space) and their corresponding semantic representations (Semantic Modelling Space). We base our definitions on the notions of layers and contexts and explain how mappings that exist between a number of modelling languages (e.g. UML and OWL) could be used to link the Business Modelling to the Semantic space. We also outline how links between layers can be used to generate bridge rules between contexts.

4.1. Business and Semantic Modelling Spaces

We consider the business modelling space, denoted BMS, as a finite non-empty countable set of modelling layers including at least one Domain-Specific Process Modeling Language layer ($dspml$), an Inter-Layer Connection Bridge layer ($ilcb$), and one BPMN layer ($bpmn$). In addition, we consider the encoded set of connections between the above layers (i.e. transformation code in Fig. 3), as an additional layer ($connect$).

Formally $BMS = \{DSPML, ilcb, bpmn, connect\}$ with $DSPML = \{dspml_i\}$, $i > 0$ since we can have more than one domain specific modelling layers as we may have different domain-specific languages. This can be useful when concepts from one domain are re-used in the modelling of other domains. For instance, when modelling the transportation of equipment (i.e. transportation domain language), health hazardous activities could also be modelled with the help of a healthcare domain specific language. We define each layer λ as $\lambda =< id_\lambda, \Gamma_\lambda, K_\lambda >$ where:

- id_λ is a unique identifier for the layer (unique in the BMS).
- Γ_λ is the language of the layer.
- K_λ is the non-empty set of expressions written using Γ_λ that represents the knowledge of the layer. We assume that K_λ is well-formed.

The semantic space, denoted SMS, is also a finite non-empty countable set of layers, each one being partially result of a transformation procedure mapping each layer of the form $\lambda \in \{DSPML, ilcb, bpmn\}$ to at least one layer in the SMS. According to the requirements outlined in Section 2, if each layer is viewed as a knowledge source, we would like to be able to treat the SMS as a set of heterogeneous knowledge sources (in terms of supported formalisms, languages) that are inter-linked, and where the representation of each link is potentially rich enough to describe the flow of information between different sources, for consistency checking and advanced querying support. In this work we view the SMS as a Multi-Context System (MCS) and each of its layers correspond to at least one MCS context. Accordingly, we define SMS as a set of contexts, $SMS = \{ctx_1, ctx_2, \ldots, ctx_n\}$, and each context is defined as a 4-tuple: $ctx_i =< id_{ctx_i}, L_{ctx_i}, kb_{ctx_i}, br_{ctx_i} >$ where

- id_{ctx_i} is a unique identifier for context ctx_i (unique in the SMS).
- L_{ctx_i} is the formal logic of the context (see Section 3) i.e. a 3-tuple $< KB_{L_{ctx_i}}, BS_{L_{ctx_i}}, ACC_{L_{ctx_i}} >$.
- kb_{ctx_i} is the knowledge base of the context such that $kb_{ctx_i} \in KB_{L_{ctx_i}}$.
- br_{ctx_i} is a set of bridge rules used to link different contexts.

4.2. Types of Mappings Considered

We deal here only with semantically-compatible mappings between the BMS and the SMS.

Semantically-compatible mapping. A mapping M is semantically-compatible iff there exists a mapping $\overline{M}$ such that $M \circ \overline{M} \subseteq I_{K_\lambda}$, where $\overline{M}$ is the inverse of M and I_{K_λ} is the interpretation of the corresponding sets of expressions.

Based on the definitions in Section 4.1 the following mappings are necessary.

- ***Id mapping:*** A function $\mathrm{i} : id_\lambda \rightarrow id_{ctx}$. This could be based on a hash function. $_M$
- ***Knowledge mapping:*** A partial mapping M between K_λ and $KB_{L_{ctx}}$ such that $K_\lambda \overset{M_\lambda}{\nrightarrow} KB_{L_{ctx}}$. Both K_λ and $KB_{L_{ctx}}$ should be well-formed.
- ***Layer-specific mapping:*** It follows that $K_\lambda \overset{M_\lambda}{\nrightarrow} kb_{ctx}$ such that $M_\lambda \in M$.
- ***Connections-specific mapping:*** A partial mapping M_{conn} between $K_{connect}$ and br_{ctx}, $K_{connect} \overset{M_{conn}}{\nrightarrow} br_{ctx}$, so that br_{ctx} is well-formed.

The above mappings should preserve well-formedness, meaning that they should guarantee some form of syntactic validity according to rules defined in the corresponding specification of the formalism. We assume that once the transformation is done, based on the above mappings, extra work can be done in the SMS to make sure that additional semantic coherence is achieved, up to the point that consistency is reached (i.e. according to the MCS theory, an equilibrium state is reached). For instance, bridge rules can be used to achieve that.

Note that a reversible (or semantics-preserving) mapping is a special case of a semantically-compatible mapping i.e. a mapping M_{SEM} is reversible or semantics-preserving iff there exists a mapping $\overline{M}_{SEM}$ such that $M_{SEM} \circ \overline{M}_{SEM} = I_{K_\lambda}$. Results of inferences or changes done in the SMS can be automatically fed back to the BMS iff the defined mapping is reversible.

4.3. Layer Specific Transformation

One of the advantages of the current method is that although modifications can occur independently in different layers, and consequently contexts, the impact of those changes can be propagated in a point-wise manner to other contexts. This means that the modeler can choose which parts of the models should be common and in what way among different layers. This of course offers much more flexibility to the modeler, however it requires defining the mapping of each layer to the corresponding context. In the rest of this section we present several examples of such mappings to illustrate that this is feasible based on existing work.

DSPML layer(s)to context(s). Defining a generic mapping for the DSPML(s) is impossible, as this depends on the corresponding formalisms defined by each DSPML. We can just note here that if UML can be used for representing those formalisms, and if they are well-formed, then the mapping presented for instance in Lagos et al.[4] could be

used. Another option is to constrain the entities ported into the corresponding context into the ones manipulated within the ILCB layer and porting them to the same semantic formalism (e.g. OWL2. This of course would mean losing some of the flexibility enabled with the current method but would still allow independent changes to be performed once the transformation is carried out and the entities are encoded in the semantic space.

ILCB layer to context. The data manipulated in the ILCB layer are modelled based on an ILCB metamodel. A lot of different ILCB metamodels may exist, depending on different needs, but the main elements required will be similar to the ones presented in Fig. 4 (note that the metamodel itself is not a contribution that we claim for this paper, rather its existence is a prerequisite (easily achieved). In fact, for this work we have experimented with a version of such a metamodel from the Eclipse Mangrove project (http://www.eclipse.org/mangrove/)). Using a mapping from the UML formalism into the formalism used in the SMS is required for the transformation. For example, if the formalism used in the context is OWL2, we could use the mapping presented in Lagos et al.[4].

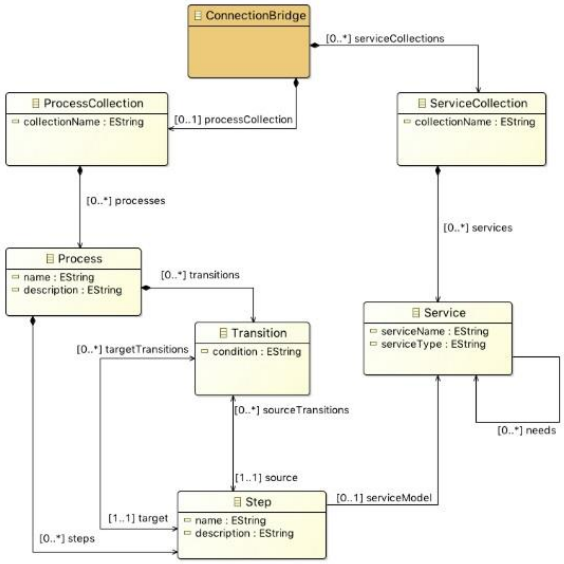

Fig. 4. Inter-Layer Connection Bridge metamodel.

BPMN layer to context. The data manipulated in the BPMN layer are based on a BPMN metamodel[2]. Using a mapping from the BPMN formalism into the formalism used in the SMS is required for the transformation, as explained in Section 4.2. For example, if the formalism used in the context is OWL2, we could use the mapping presented in Rospocher et al.[12].

Connections to bridge rules. To connect the different contexts, we use the notion of bridge rules as explained in previous sections. Bridge rules representing explicitly the derivation of an object in one layer from another object(s) in other layers, can be generated in accordance with the transformation code defined among layers i.e. a transformation is carried out from the DSPML to the ILCB layer thanks to some code (see Fig. 3). In the case of existing MBPM systems the link between the two objects is then lost (except for potentially implicit information used in the naming of the resource). In our case, this link can be encoded as an explicit bridge rule when the transformation to the semantic space takes place, relying on information already encoded in the transformation code (e.g. in Java code). For instance, if we assume that context 1 or dsl is the name of the context that corresponds to the DSPML and context 2 or $ilcb$ the name of the context that corresponds to the ILCB layer, bridge rule br_1 could be used to represent that an entity in dsl is linked to an entity in $ilcb$:

$$\exists a_1, a_2 \text{ such that } br_1 = \{(2 : a_2) \leftarrow (1 : a_1)\}$$

where a_1 and a_2 represent atoms that relate the object from the DSPML layer to the corresponding object generated in the ILCB layer. Operationally, we would have to add some code to instantiate those rules every time such an operation happens, which should be straightforward. In this example, br_1 expresses a non-symmetric inference, however according to the formalization used and the result desired, different/more expressive bridge rules

can be used. For instance, br_2 checks if a_1 and a_2 are typed in their corresponding contexts and every time that the generation operation is performed in the BMS, the corresponding rule is fired to generate an extra triple in the $ilcb$ context (we assume that the context is encoded as a relational database and the one in RDF).

$\exists a_1, a_2$ such that $br_2 = \{2:(a_1\ skos:relatedMatch\ a_2) \leftarrow (1:\text{type}(a_1, ?\text{type}1)), (2:a_2\ rdf:type\ ?type2)\}$

? denotes a variable (stands for all respective instances obtainable by value substitution). The triple links the two entities using the skos:relatedMatch predicate (skos:relatedMatch is used to state an associative mapping link between two SKOS entities (http://www.w3.org/TR/2009/REC-skos-reference-20090818/)). Note that we assume that an IRI is used for the naming of a_1 in the $ilcb$ context. Other types of bridge rules, more complex, to represent richer relations among entities from different contexts could also be defined.

Manually added knowledge. In addition to the information available from the automatic transformation, other definitions can be inserted in each of the contexts of the SMS (it makes sense here to allow only users with appropriate profiles/rights to be able to add such definitions). The definitions have to be expressed using the appropriate supported formalism. For instance, in the case of an RDF-based triple store the following fact could be inserted to represent the temporal ordering of two tasks:

```
ex:Payment ex:after ex:InvoiceProcess
```

meaning that all payment operations must be done after processing invoices corresponding to the operation. The binary predicate ex:after could have rich semantic properties attached to it, such as transitivity, which would be used by a reasoning engine to infer additional information.

5. Responding to the Challenges: Revisiting the Example

In this section, we discuss how the framework presented in this paper could be used to resolve the two key challenges identified in Section 2: *cross layer consistency* and *layer focused querying*.

Resolving Challenge 1: Keep Information Consistent Across Layers. As illustrated in Section 2, a typical modification that can take place in the BPMN layer is to change the type of an object. In our example, the type of `Process Disease Indicators` is changed from `Subprocess` to `Service` (Fig. 2). The initial type defined in the DSPML and propagated to the ILCB layer is still `Subprocess`. In that case, the following pseudo-code denotes the corresponding information in the ILCB layer:

```
Process_Disease_Indicators isA Process
```

We have the following semantically-compatible mappings for the two layers (X, Y stand for variables and :– denotes a production rule, on the left being the head):

Table 1. Knowledge mappings.

ILCB	X isA Y	$Y :- X$
BPMN	L K	$L :- K$

Table 2. Layer-specific mappings.

ILCB	Process_Disease_Indicators isA Process	Process:-Process_Disease_Indicators.
BPMN	ServiceTask Process Disease Indicators	ServiceTask:-Process_Disease_Indicators.

We generate, minimally, bridge rules, as defined in Section 4.3, that link two objects with the same naming in the two different layers (this minimally exists to allow the connection to the ILCB layer). This means that we have the following bridge rules generated (Context 1 corresponds to the ILCB one and context 2 to the BPMN one. The direction of the rules is chronologically defined i.e. the left part of the rule includes the context in which the object was last changed.), and their inverse as well to express the equivalence:

```
(1:Process):-(2:Process).
(2:ServiceTask) :-(1:ServiceTask).
(1:Process_Disease_Indicators):-(2:Process_Disease_Indicators).
```

The above mappings give us the following Answer-Set Program (executed using the mcs-ie tool[13]):

```
#context(1,"dlv_asp_context_acc", "ilcb.dlv").
#context(2,"dlv_asp_context_acc", "bpmn.dlv").
```

```
r1: (1:process_disease_indicators) :- (2:process_disease_indicators).
r2: (2:process_disease_indicators) :- (1:process_disease_indicators).
r3: (1:process) :- (2:process).
r4: (2:process) :- (1:process).
r5: (1:service_task) :- (2:service_task).
r6: (2:service_task) :- (1:service_task).
===ilcb.dlv===
process :- process_disease_indicators.
:- process, service_task.
===bpmn.dlv===
service_task :- process_disease_indicators.
```

with one of the following equilibria:

```
({process,process_disease_indicators,service_task},{process,process_disease_indicators,service_ta
sk})
```

which denotes that for both contexts all three objects are included in a valid belief set.

However, according to the semantics of the ILCB metamodel (Fig. 4) we denote that the classes `Process` and `ServiceTask` do not have any intersection since the Process and Step classes are disjoint in the ILCB metamodel (they do not share a common ancestor). Although not shown in Fig. 4, Subprocess is a subclass of Process. Each ServiceTask instance is a Step instance with an association to a Service instance. We denote this manually (or automatically if it is defined in the mappings) in the ILCB context as follows:

```
:- process, serviceTask.
```

Executing the new program results in only the following belief sets being acceptable:

```
({process},{process})
({service_task},{service_task})
```

meaning that only one of the above objects can be included at the same time in a valid belief set for both contexts. This raises an inconsistency, as we require `process_disease_indicators` to be part of the valid belief sets.

Resolving Challenge 2: Support Querying for Different Layers. As discussed in the example of Section 2, we may need knowledge bases expressed in different formalisms to be able to find out that measles is a viral infection and to also infer that we are exceeding a specific time limit e.g. 30 minutes. To make the link to the corresponding knowledge bases we define the links in the DSPML. For instance, the mappings to the Disease Ontology (DO) and SNOMED-CT are shown in Fig. 1(a) at two different places – at the Disease Clinical Pathway DSPML at the Update Medical record concept and in the Disease Clinic Evaluation DSPML at the enum RULE_TYPE field. The mapping to the OWL-Time ontology is shown in the TIME field. The links in this case are denoted as an IRI, as ontologies are based on the notion of IRIs for unique identification. However, to be able to query both knowledge bases in the same framework, the corresponding reasoning strategies should be supported.

SNOMED-CT, because of its sheer size, is restricted to the EL++ DL expressivity[14], which is one of the few description logics for which standard reasoning problems such as ontology consistency, concept subsumption, and instance checking are decidable in polynomial time. To achieve that performance, commonly-used constructors such as universal value restrictions and inverse and functional roles are disallowed. For SNOMED-CT though, scalable reasoning is more important than the lost language expressivity[15].

OWL-Time contains a lot of commonly used notions for temporal inference such as the *before* and *after* properties that can be used to represent time sequences i.e. that a time point or interval is after/before another time point interval. What is even more interesting for our example is that it explicitly defines the semantics of a minute, in relation to intervals, durations and other temporal units e.g. second or days. This would allow us to respond to a query involving a comparison of temporal units in relation to a duration entity like the one above. However, OWL-Time includes a number of concepts that require a higher expressivity than EL++.

It is obvious that in order to be able to respond to Question 1 and Question 2 (c.f. Section 2) we need access to both of the above knowledge bases. We can achieve that in our framework by denoting two different contexts for each of the two knowledge bases e.g. *DO* and *Time*. Query answering in our context would include executing two different queries one on the *DO* context and one on the *Time* one. In this paper, we do not provide more details on how the corresponding queries can be done, but this can be straightforward under specific conditions if each context is queried independently (SPARQL, the RDF query standard language could be used). Research work on queries involving multiple contexts is an active research field.

6. Related Work

We are not aware of any work proposing a mapping of multi-layered representations of business processes to the multi-context systems framework. However, a lot of work around adding semantics to Business Process Models has been carried out over the last years. For instance, Mendling et al.[16] have proposed exploiting the textual content found in business process models to enable semantic-based change management and querying tasks. We believe that this work is complementary to ours. In the domain of Semantic Business Process Management ontologies have been extensively used to models parts of processes and artefacts and enable querying[17]. The closest work in that vain, is that of Mayer et al.[18] who suggests modeling the interests of different stakeholders, by allowing the use of a core upper ontology that captures the main characteristics of a process and can be extended according to different requirements by other domain-specific ontologies. However, the key idea of harmonizing different representations of semantically similar objects (those differences being the consequences of technical artefacts and possibly uncontrolled/decoupled evolution processes) and dealing with the various knowledge levels and associated languages is not addressed.

Maintaining consistency among business process models represented at different abstraction levels includes several aspects: functionality, information density (e.g. granularity of modelling), and behavior (e.g. how the execution sequence of corresponding tasks is implemented)[19]. In all cases, a notion of equivalence is defined so that inconsistency can be identified. For instance, Yongchareon et al.[20] formally defines under which conditions behavioral consistency can be preserved when specializing processes in a synchronized manner (i.e. extending or refining them). Especially interesting is the notion of behavioral profiles, which define the concepts of mutual exclusion, and strict and interleaving order for activities at different abstraction levels[21]. As defined in the previous sections, inconsistency in our case is defined minimally as the lack of an equilibrium or as a refined set of acceptable belief sets, and therefore support the view that consistency is a subjective notion and the corresponding rules should be customizable[22]. Some of the above works could be used to define belief sets in our framework.

Approaches for business process model querying have been proposed by several researchers either based on specialized query frameworks[23] or appropriate ontologies[24]. We do not propose a new query language or method but present how layer and context-specific querying, in different languages, could be supported in our framework. Enabling cross-layer querying would be even more interesting in the future and frameworks such as PQL[25] and RMCS[26] will be explored further.

In earlier work[4], we informally introduced the idea of externalizing the semantics of the domain-specific and bpmn models in a separate space, exemplified via the use of ontologies. However, the key notion of allowing the co-existence of different formalisms and corresponding tools in the same framework, was not presented. In addition, formalizing and realizing the above based on the MCS framework was not discussed.

7. Summary and Conclusion

We have presented in this work a new method and system that transforms business models into semantic models. This allows inferences to be performed for conflict identification and supports query execution. To achieve the transformation, we have defined the notion of a business modelling space as an aggregation of layers that hold domain-specific and BPMN related representations, in addition to the connections between those two views represented as an intermediate layer. A method to transform knowledge held in such a system into a multi-context system has then been defined.

The advantages of the framework introduced in this paper when compared to previous approaches include: 1. identification of conflicts/inconsistencies across different layers; 2. possibility of expressing rich information on the relations between two contexts thanks to the notion of bridge rules; and 3. calculation of the impact of changes in one layer to the rest of the layers.

The potential drawback, namely, the additional complexity of defining the map-ping from the BPM space to the semantic space, has been addressed through an example suggesting that the needed knowledge is partly pre-existing during the model design phase and can also be partly extracted from pre-existing interlayer translation code. In addition, existing work on language mappings (e.g. UML to OWL) could be used.

We consider this work as one of the foundational bricks that will enable further advances towards the governance of multi-layer business process modelling systems, as we anticipate that the semantic mechanisms introduced here can be further enriched, for instance, in order to embed semantic bridges towards the IT layer (hence enabling search and inference mechanisms not restricted to business concepts but involving business concepts and IT capabilities altogether). Another path towards reinforcing the governance capabilities based on this approach would consist in enriching the proposed inter-layer connection bridge with execution information so as to consider evolutions in the overall business description enabled by this work.

References

1. Mos A. Domain Specific Monitoring of Business Processes using Concept Probes. Intelligent Service Clouds Workshop (ICSOC 2014). Volume: *ICSOC 2014 workshops and satellite events*, Nov 3 - Nov 6, Paris, France (2014)
2. Object Management Group, Business Process Model and Notation, http://www.bpmn.org/
3. Mos A, Jacquin, T. Improving Process Robustness through Domain-Specific Model Transformations. *Enterprise Distributed Obj. Comp. Conf. Workshops* (EDOCW 2013), Vancouver, BC (2013)
4. Lagos N, Mos A, Vion-Dury JY, Chanod JP. Preserving Consistency in Domain-Specific Business Processes through Semantic Representation of Artefacts. 7th Work. on Applications of Knowledge-Based Tech. in Business (AKTB 2015). In: Abramowicz, W. (eds.). *BIS 2015 Int. Workshops*, Poznań, Poland, June 24-26, 2015, Revised Papers. LNBIP, 228 36-47. Springer, Heidelberg (2015)
5. https://www.nlm.nih.gov/healthit/snomedct/index.html
6. http://disease-ontology.org/
7. https://www.w3.org/TR/owl-time/
8. Brewka G, Eiter T. Equilibria in heterogeneous nonmonotonic multi-context systems. In Cohn, A. (eds.), Proc. *22nd Conf. on Art. Intell. - Volume 1* (AAAI'07), AAAI Press 385-390 (2007). https://www.aaai.org/Papers/AAAI/2007/AAAI07-060.pdf . Accessed: 21 Mar 2016
9. Brewka G, Eiter T, Fink M, Weinzierl A. Managed multi-context systems. *Intern. Joint Conf. on Artificial Intelligence* (IJCAI 2011), 786-791 (2011). http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.297.8409&rep=rep1&type=pdf . Accessed: 21 Mar 2016
10. Eiter T, Fink M, Schuller P, Weinzierl A. Finding explanations of inconsistency in multi-context systems. *Principles of Knowl. Repres. and Reas.: Proc. Twelfth Int. Conf.* (KR 2010), Toronto, Ontario, Canada, May 9-13, AAAI Press (2010)
11. Barilaro R, Fink M, Ricca F, Terracina, G. Towards Query Answering in Relational MultiContext Systems. In Cabalar and Son, (Eds.), *Proc. 12th Intern. Conf. on Logic Programming and Nonmonotonic Reasoning* (LPNMR 2013), Corunna, Spain, September 15-19, 2013, LNCS, 8148, 168–173. Springer (2013)
12. Rospocher M, Ghidini C, Serafsini L. An ontology for the Business Process Modelling Notation. *Formal Ontology in Inf. Sys. Proc. 8th Int. Conf.* (FOIS2014) September, 22-25, 2014, Rio de Janeiro, Brazil, vol. 267, pp. 133-146, IOS Press (2014)
13. Bögl M, Eiter T, Fink M, Schüller P. The mcs-ie System for Explaining Inconsistency in Multi-Context Systems. *Logics in Art. Intell*. In Janhunen, T., Niemelä, I. (Eds.) Vol. 6341, Berlin, Heidelberg: Springer Berlin Heidelberg, 356–359 (2010)
14. Baader F, Brandt S, Lutz C. Pushing the EL Envelope Further. In *Proc. (OWLED 2008) DC Work. on OWL: Exper. and Direct.* (2008)
15. Dentler K, Cornet R, ten Teije A, de Keizer N. Comparison of reasoners for large ontologies in the OWL 2 EL profile. *Sem. Web* 2(2) 71-87 (2011)
16. Mendling J, Leopold H, Pittke F. 25 Challenges of Semantic Process Modeling. *Intern. J. Inform. Sys. and Soft. Eng. for Big Companies* (IJISEBC) 1, 78–94 (2015)
17. Hepp M, Leymann F, Domingue J, Wahler A, Fensel D. Semantic Business Process Management: A Vision Towards Using Semantic Web Services for Business Process Management. *Proc. IEEE ICEBE 2005*, October 18-20, Beijing, China, 535-540 (2005)
18. Mayer R, Antunes G, Caetano A, Bakhshandeh M, Rauber A, Borbinha J. Using ontologies to capture the semantics of a (business) process for digital preservation. *Intern. J. of Dig. Lib.* (IJDL), 15 129-152 (2015)
19. Weidlich M, Barros AP, Mendling J, Weske M. Vertical alignment of process models - how can we get there? In *CAiSE 2009 Work. Proc.: BPMDS*, 71–84 (2009)
20. Yongchareon S, Liu C, Zhao X. A Framework for Behavior-Consistent Specialization of Artifact-Centric Business Processes, in: Barros, A., Gal, A., Kindler, E. (Eds.), *Business Process Management*. Springer Berlin Heidelberg, Berlin, Heidelberg, 285–301 (2012)
21. Weidlich M, Mendling J, Weske M. Efficient consistency measurement based on behavioural profiles of process models. *IEEE Trans. on Soft. Eng*. 99(PrePrints) (2010)
22. Branco M. Managing Consistency of Business Process Models across Abstraction Levels. University of Waterloo, Waterloo, ON, Canada (2014). http://hdl.handle.net/10012/8310. Accessed: 21 Mar 2016
23. Beeri C, Eyal A, Kamenkovich S, Milo T. Querying business processes with BP-QL. *Inf. Syst.* 33(6) 477–507 (2008)
24. Smith F, Missikoff M, Proietti M. Ontology-Based Querying of Composite Services, in: Ardagna CA, Damiani E, Maciaszek LA, Missikoff M, Parkin M. (Eds.), *Business System Management and Engineering*. Springer Berlin Heidelberg, Berlin, Heidelberg, 159-180 (2012)
25. Polyvyanyy A, Corno L, Conforti R, Raboczi S, La Rosa M, Fortino G. Process Querying in Apromore. *Proc. Demo Track 13th Intern. Conf. on Business Process Management* (BPM'15) (2015)
26. Barilaro R, Fink M, Ricca F, Terracina G. Towards Query Answering in Relational Multi-Context Systems, in: Cabalar, P., Son, T.C. (Eds.), *Logic Programming and Nonmonotonic Reasoning*. Springer Berlin Heidelberg, Berlin, Heidelberg, 168–173 (2013)