

Using Ontologies for Business Capability modelling: Describing What Services and Processes Achieve

WASSIM DERGUECH^{1*}, SAMI BHIRI² AND EDWARD CURRY¹

¹*Insight Centre for Data Analytics, National University of Ireland, Galway, Ireland*

²*OASIS, National Engineering School of Tunis, University Tunis El Manar, Tunisia*

*Corresponding author: wassim.derguech@insight-centre.org

In current business process modelling environments, the functional perspective (also can be referred to in the literature as business capability, functionality or business function) for each process activity is limited to its label. Using labels only prevents stakeholders from easily and quickly understanding what business processes or services achieve. In this paper, we define a business capability meta-model that can be used for modelling business capabilities as entities composed of a set of actions and related properties. This meta-model is implemented as Resource Description Framework (RDF) vocabularies that help experts design their domain-specific high-level business capabilities that can be used for annotating their processes, services, applications, etc. We validate the business capability meta-model by using two evaluation methods: (1) ontological evaluation in order to make sure that there is no semantic ambiguity among its constructs; (2) interviews with domain experts to assess the ability of the model to represent real capabilities and to evaluate their point of view with respect to our approach.

Keywords: business capability; business process; service; RDF; semantic web

Received 24 July 2017; revised 11 October 2017; editorial decision 9 January 2018

Handling editor: Rada Chirkova

1. INTRODUCTION

In current business process models, the functional perspective (also can be referred to in the literature as business capability, functionality or business function) for each process activity is limited to its label [1]. A single label is not enough to describe properly the capability of a particular process element (i.e. activity, fragment or entire process). Using labels only prevents stakeholders from easily and quickly understanding business processes or identifying the differences and commonalities between them in terms of business properties [2]. When required, stakeholders need to read the business process documentation in order to find out what a process element does, expressed in terms of business properties.

Information Systems' vendors such as IBM, Oracle or SAP offer together with their solutions the related documentation that is usually (1) extremely large and (2) combines various levels of the technical implementation [3]. For example, searching in SAP ERP documentation requires in depth knowledge of a large and proprietary terminology [3]. This problem can be resolved

by properly describing capabilities in a way that serves machine processing (well-structured capabilities allow for their indexing, searching, detecting differences between them, etc.) and featuring business rather than technical terms (business properties that business experts are familiar with). Therefore, the problem that we are addressing in this paper is the lack of a structured business capability model that allows to describe what services, processes, computer applications, etc. do from a functional perspective.

This problem has been investigated for the modelling of IT capabilities. Indeed, the literature proposes various IT capability description approaches as a part of efforts for describing related concepts such as business processes, services and search requests (WSMO [4], OWL-S [5], SA-WSDL and SA-REST [6, 7]). They primarily describe capabilities either as part of their implementations (i.e. invocation interface) or as part of other concepts (i.e. services). For all these approaches, the semantics of the action performed by services are derived through reasoning over its inputs, outputs, preconditions and effects (IOPE). The main criticism towards these approaches

comes from the fact that they mainly focus on modelling IT capabilities rather than business capabilities.

Oaks *et al.* [8] explores a frame-based modelling approach for describing service capabilities by using natural language constructs such as the action performed and associated parameters (i.e. temporal, location, etc.). Even though the action performed is captured in terms of action verbs, the associated business parameters remain as a part of the service inputs, outputs, preconditions and effects (IOPE). As the solution proposed by Oaks *et al.* does not go beyond classical IOPE-based capability descriptions (i.e. IT capabilities), we consider that it has the same problem of semantic web service solutions.

In this paper, and more specifically in Section 3, we define a conceptual model for describing the actions of services and processes in a structured format. We use the term *Business Capability* to refer to this functional perspective [8]. The model should define business capabilities as *standalone entities* independent from their implementations and *feature business terms*. Business Capabilities should be *machine processable*: can be *indexed*, *searched* and *compared* between each other. Business Capabilities and related concepts should be defined by domain experts (and optionally by modelling experts) and presented in domain-related ontologies (that we call capability domain ontologies) and serialized in a standard format to facilitate their portability.

In terms of realization, in Section 4, we have been particularly interested in investigating the use of semantic web technologies for implementing the model as a set of vocabularies using the Resource Description Framework (i.e. aka RDF). These vocabularies/ontologies can be used to create semantic annotations of services or processes, an approach that is widely used in semantic web service management [9]. Its main advantage is that we create semantic annotations without constraints from the service or process modelling language; instead, we only need hooks in services or process descriptions where semantic annotations can be attached. This choice is further motivated by the vision of better enabling *computers* and *people* to work in cooperation [10]. Within this vision and in the context of business process modelling, *people* represent *business experts* that are expressing their needs and requirements in terms of *Business Capabilities* and describing applications and processes from a *functional perspective*.

We provide a tool support, discussed in Section 4.6, that allows stakeholders to annotate a particular process model using predefined business capabilities from a *capability domain ontology*. This requires the extension of the chosen business process modelling language serialization to integrate business capability descriptions. We call the resulting models *Business Capability Annotated Business Process Models*.

To evaluate the proposed model, we used two methods in Section 6:

- (1) *Ontological Evaluation*: The ontological evaluation of conceptual models consists of mapping the proposed

conceptual model constructs to ontological concepts/constructs in order to assess the ability to model capabilities without semantic ambiguity by avoiding construct overload and redundancy [11, 12].

- (2) *Interviews with Domain Experts*: We chose to carry out semi-structured interviews [13, 14] with five domain experts that have strong background and are currently active in the area of service computing and information system design and development. The interviews were done after explanation of the objective of this work and details about service modelling approaches. The main targeted outcome of these interviews was to identify if these experts can confirm that the proposed model is good enough to model business capabilities and if it can be adopted in their working environment.

Before concluding the paper and defining future perspectives in Section 8, we discuss the related work in Section 7 with respect to a set of requirements that are defined in Section 2.

2. DEFINITIONS AND REQUIREMENTS AS UNITS OF ANALYSIS

The concept of capability has been defined by Dutta *et al.* [15] and Amit and Schoemaker [16], from organizational and data perspectives, as the ability of organizations to efficiently use their resources (i.e. human capital, knowledge, available data, etc.) to generate value and achieve their objectives. BPM [17], defines capabilities, from a control-flow perspective, as the way (i.e. HOW) organizations achieve their goals by capturing explicitly process tasks and their temporal and logical order. From a functional perspective, OASIS Reference Model [18] considers capabilities as the effect of a service in terms of data generated or change of the world and Oaks *et al.* [8] define capabilities as what a service does in terms of action performed that creates a value for the customers.

Considering the functional perspective, the definition given by OASIS Reference Model [18] is tight to the actual implementation of a service and thus defines *IT capabilities* while Oaks *et al.* [8] try to give a more *business/functional capability* definition. The capability conceptual model proposed by Oaks *et al.* [8] uses IT capabilities of services for deriving the semantics of the action performed. Furthermore, OASIS Reference Model [18] considers the concept of a service as a core element that enables a requester to access and achieve a particular *business capability*. Within this vision, we notice that the concept of service has evolved from the notion of remote invocation interface (such as WSDL [19]) to a more comprehensive entity. Within this vision, the invocation interface is only one aspect of the whole service description.

Another core aspect of a service, which is the focus of this paper, is the notion of *business capability* which describes what a service achieves.

The notion of business capability is a fundamental concept not only for Service-Oriented Architecture (SOA) but also for enterprise information systems. The ARIS architecture [20] recognizes the importance of the functional perspective in enterprise information systems and considers it as one of its views. The concept of business capability is the glue point between services and business processes. A service gives access to a certain capability which can be achieved by a business process. Despite its importance, this concept has not drawn the research community attention as it deserves. Current approaches for capability modelling were part of efforts for describing related concepts such as business processes, service descriptions and search requests.

Sycara *et al.* identified a set of high-level requirements that a capability description language should consider when modelling software agents' capabilities that can also be adopted in the context of this work (i.e. services and business processes):

- *Requirement 1: Expressiveness*—A business capability modelling language should be expressive enough to represent the meaning or the action behind the actual capability. Action's semantics should be explicitly defined and not relying on inferences and analysis of its effect. Furthermore, capabilities should be described independently from their implementations. This requirement was elicited from the following works: Sycara *et al.* [21], Oaks *et al.* [8], Semantic Web Services Models: WSMO [22] and OWL-S [23], and Semantic Annotation of Invocation Interfaces Models: SA-WSDL [24, 25] and SA-REST [26]. This requirement can be further refined as follows:
 - *Requirement 1.1*—Explicitly represent the action performed.
 - *Requirement 1.2*—Explicitly capturing functional and non-functional features related to the action performed.
 - *Requirement 1.3*—Ability to express these features using simple (e.g. integer, boolean, string) as well as complex types (e.g. conditional values, enumerations).
- *Requirement 2: Inferences*—Given a set of business capability descriptions, additional knowledge can be inferred such as identification of relationships between business capabilities and indexing. Such features can be used to efficiently discover and compose capabilities. This requirement was elicited from the following works: Sycara *et al.* [21], Oaks *et al.* [8], and Semantic Web Services Models: WSMO [22] and OWL-S [23]. This requirement can be further refined as follows:

- *Requirement 2.1*—Ability to explicitly identify relationships between capabilities based on their descriptions.

Requirement 2.2—Ability to index and search capabilities.

- *Requirement 3: Use of Ontologies*—Describing business capabilities requires sharing of common understanding of the structure of this information and enable the reuse of its constructs among the involved stakeholders. In such context, the use of ontologies is key enablers. A business capability description language should support the use of domain and common ontologies for specifying capabilities [21]. This requirement was elicited from the following works: Sycara *et al.* [21], Oaks *et al.* [8], Semantic Web Services Models: WSMO [22] and OWL-S [23], and Semantic Annotation of Invocation Interfaces Models: SA-WSDL [24, 25] and SA-REST [26]. This requirement can be further refined as follows:

- *Requirement 3.1*—Use of domain or general ontological concepts for describing capabilities.
- *Requirement 3.2*—Searching capabilities should not be relying on keyword extraction and comparison.

We use these requirements in the rest of the paper for driving our design decisions in Sections 3 and 4 as well as conducting a related work analysis in Section 7.

3. CAPABILITY META-MODEL

We propose in this section our contribution as a business capability meta-model. This meta-model defines the high-level concepts required for defining domain-specific business capabilities.

3.1. Business Capabilities as Property-featured Entities

We propose to model a business capability as an *action category* enriched by (zero or many) *functional or non-functional properties* (see the concept of property entry below). As examples, we refer to Table 1¹ that lists five capabilities with the same action category: i.e. 'Shipping'. *Capability A* has two functional properties: i.e. *From* and *To* with *International Address* as possible values. Simply, *Capability A* describes the shipping action within two international locations. The rest of the capabilities, in the same table, have either additional properties or different property values.

¹Note that the notation used in these examples is not formal, formal descriptions of the used concepts is shown in Section 4.

TABLE 1. Examples of Business Capabilities.

Capability	Action category	Properties	Property value/ possible values
Capability A	Shipping	From To	International Address International Address
Capability B	Shipping	From To	International Address International Address
Capability C	Shipping	From To MaxWeight	International Address International Address 68 kg
Capability D	Shipping	From To MaxWeight	European Address European Address 1 t
Capability E	Shipping	From To MaxWeight PickUpDate	European Address European Address 68 kg Date

More formally, in the proposed model, business capabilities are defined as a *Category* and a set of *property entries* (see Definition 2). A *property entry* as described in Definition 1 (see also Example 1) is a couple (*property*, *value*) where *property* is a *functional* or a *non-functional property* and *value* is the value or the possible values that a property can have. Both *property* and *value* refer to ontological terms.

DEFINITION 1. *Property Entry, Property Declaration and variantOF*

- A property entry (P, v) is specified w.r.t. a property declaration defined in a shared ontology. As example, we refer to columns 3 and 4 of Table 1. Column 3 shows the list of properties P and column 4 shows their respective values v .
- A property declaration, $d = (P, V, R)$, defines (i) a property P as a relevant functional or non-functional feature of the capabilities of a given domain, (ii) V the most general value (super class) that property entries defined according to d can have and (iii) a set of relations R that tell when a value v_1 is more specific than a value v_2 w.r.t. the semantics/meaning of the property P (see Example 1).
- Let v_1 and v_2 be two values and R a set of specification relations. v_1 variantOF v_2 if $\exists r \in R$ such that $v_1 \text{ } r \text{ } v_2$ (see Example 1).

EXAMPLE 1. *Property Entry and Property Declaration.*

Capability A, listed in Table 1, includes as part of its description the property *From* that indicates where (location)

TABLE 2. Examples of Property Declarations.

Property declaration	Property name	Most general value	Specification relation(s)
d_{From}	From	GeographicalLocation	LocatedIn
d_{To}	To	GeographicalLocation	LocatedIn

the shipment is taken from. This property is defined w.r.t. the property declaration listed in Table 2. $d_{From} = (From, GeographicalLocation, LocatedIn)$ where *From* is the actual property, *GeographicalLocation* is the most general value of this property and *LocatedIn* is the specification relation that exists between possible values of this property.

Let $From_{EU}$ and $From_{IE}$ be two property entries. $From_{EU} = (From, Europe)$ and $From_{IE} = (From, Ireland)$ defined w.r.t. d_{From} (see Table 2). Note that the specification relation *LocatedIn* holds between the values *Ireland* and *Europe*.

DEFINITION 2. *Business Capability*

A couple $Cap = (Category, Properties)$ is a business capability, where:

- **Category:** A predefined concept in a domain-related ontology that comes from a shared agreement on its action semantics. A category is a specific property that is present in all capability descriptions via the property **achieves**.
- **Properties:** A set of property entries (Property, Value) as explained in Definition 1.

In order to give an object-oriented conceptual view of the proposed model, we refer to the UML class diagram shown in Fig. 1. This diagram contains the following classes:

- **Capability** is the class that captures the capability (see Definition 2). This class is composed of at least 1 *Action Category* and, optionally, multiple *Property Entries*.
- **Action Category** is the class that represents the category of the capability (see Definition 2).
- **Property Entry** is the class that represents a property entry that has a name and a *Value* (see Definition 1). A property Entry is defined with respect to a *Property Declaration* (the connection between both classes the same property name).
- **Property Declaration** is the class that represents the property declaration (see Definition 1). It has a property name, the most general *value* and a set of specification relations (see Definition 1).
- **Value** is the class that represents the value of a property (see Definition 1). We create a separate class for

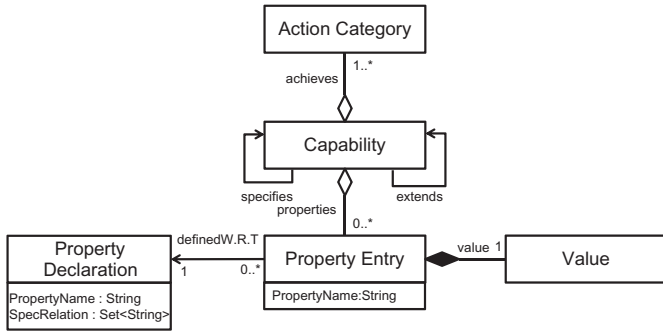


FIGURE 1. Capability UML class diagram.

values as we need to create complex types that are depicted in Fig. 2. These types are:

- *EnumerationValue* extends the class *Value* to represent enumerations of other values (see relation *hasElement* in Fig. 2).
- *RangeValue* extends the class *Value* to represent ranges of other values presented as minimum and maximum (see relations *hasMin* and *hasMax* in Fig. 2).
- *DynamicValue* extends the class *Value* to represent a dynamic value that is evaluated with respect to an Expression (see relation *hasEvaluator* in Fig. 2).
- *ConstrainedValue* extends the class *Value*, it has a value only if a certain *Constraint* is valid (see relation *constrainedBy* in Fig. 2).
- *ConditionalValue* extends the class *Value*, it has a value only if a certain *Constraint* is valid (see relation *hasCondition* in Fig. 2). Additionally, its value is computed dynamically with respect to an Expression (see relation *hasEvaluator* in Fig. 2).
- *Constraint* is the class that represents a constraint that is defined via an *Expression* (see relation *hasExpression* in Fig. 2).
- *Expression* is the class that represents an expression, it has a type and a value (both of type String).

The idea of modelling business capabilities as a set of features was highly influenced by the frame-based modelling paradigm. Indeed, the conclusions of Wickler [27, 28] after an extensive analysis of multiple modelling mechanisms and languages suggest that frame-based descriptions of capabilities in the context of software agents were the most expressive and flexible means. Additionally, using the model suggested in this paper, one can describe business capabilities independently from their actual implementations by highlighting the action being performed with a set of related properties. Contrary to the Input, Output, Precondition and Effect paradigm, it features the functional (business) and non-functional characteristics which end-users are mostly interested in and which are

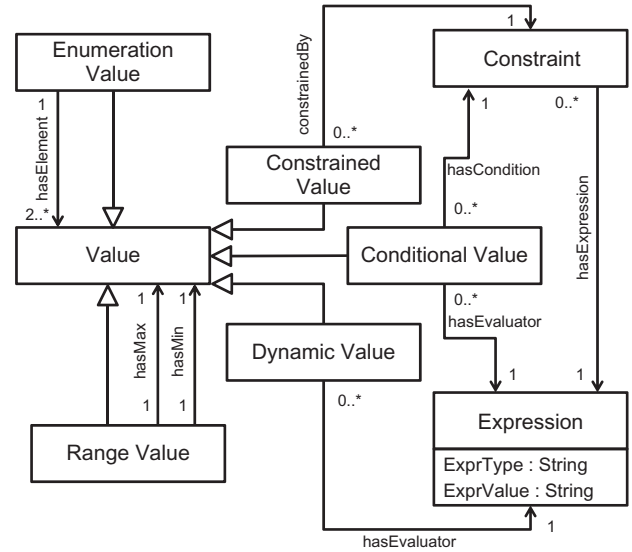


FIGURE 2. UML class diagram for the possible values.

specified in their requests. This constitutes a natural way on how users describe their needs, for example, users need a *service that ships packages from an address to another*. This is expressed via *Capability A* in Table 1 with an action category ‘shipping’ and the properties ‘from’ and ‘to’.

3.2. Specification and Extension Relations between Capabilities

Another benefit of using frame-based modelling of capabilities is the possibility to infer potential relationships between them by analysing their features (or properties). We have been particularly interested in this work in relations of *specifications* and *extensions* that may exist between capabilities. These relations are captured in Fig. 1 as ‘*specifies*’ and ‘*extends*’ between capabilities and are described, respectively, in Definitions 3 and 4.

Note:

- For abbreviation purposes and by misnomer we say that a certain capability *cap* has a property *pr*.
- We refer to the property *pr* of *cap* by *cap.pr*.
- We refer to the set of properties of *cap* by *cap.properties*.
- We say that two capabilities C_1 and C_2 (or more) share the same property *pr* if both of them have the property *pr* (but possibly with different values).

DEFINITION 3. *specifies*

Given two capabilities C_1 and C_2 , C_1 specifies C_2 if (i) all the properties of C_2 are also properties of C_1 (in other terms C_1 inherits all the properties defined in C_2), (ii) for

every shared property pr , the value of $C_1.pr$ is either equal to or variantOf the value of $C_2.pr$, and (iii) there exists at least one shared property pr' such that the value of $C_1.pr'$ variantOf $C_2.pr'$ (see variantOf in Definition 1).

DEFINITION 4. *extends*

Given two capabilities C_1 and C_2 , C_1 extends C_2 if (i) C_1 has all the properties of C_2 and has additional properties, and (ii) for every shared property pr , the value of $C_1.pr$ is equal to the value of $C_2.pr$.

Fine-grained relations can be defined based in these relations. Let C_1 and C_2 be two capabilities such that C_2 specifies C_1 , and let pr be a shared property, we say that C_2 specifies C_1 on pr , denoted C_2 specifies_{pr} C_1 iff the value of $C_2.pr$ is variantOf the value of $C_1.pr$.

The relations *specifies* and *extends* (fine or coarse-grained) enable organizing a repository of capabilities as a hierarchy [29, 30] as shown in Example 2 and illustrated in Figs 3 and 4.

EXAMPLE 2. *Hierarchy of Capabilities*

Figures 3 and 4 show two examples hierarchies of capability descriptions.² *Capability A* is the root of these hierarchies, it represents an abstract capability description for shipping goods from any source and destination at an international scale. This capability can be extended either to *Capability B* or *Capability C*. Both extend the initial capability by one or two attributes; fine-grained relations can be seen in Fig. 4. As an example of specification relation between capabilities, we refer to the link between *Capability D* and *Capability B* in Fig. 3. *Capability D* specifies *Capability B* as it becomes a European shipping capability instead of International. This fine-grained semantics of the specification relation is further shown in Fig. 4. It is also clear that *Capability E* extends *Capability D*.

Note that the hierarchies depicted in Figs 3 and 4 are simple and can be easily created manually. However, when it comes to large set of capabilities, more dedicated algorithms for creating optimal hierarchies are needed [30]. We have explored the potential of using this model to describe and create an indexing structure of sensor capabilities using Formal Concept Analysis [31]. The results show that using this approach we were able to index and discover capabilities of a large repository in few milliseconds that proves to be more time efficient than existing approaches.

The proposed conceptual model allows to model high-level capabilities in a particular domain that can be tailored to specific use cases. Similar to domain ontologies which define shared concepts and shared attributes/properties, high-level

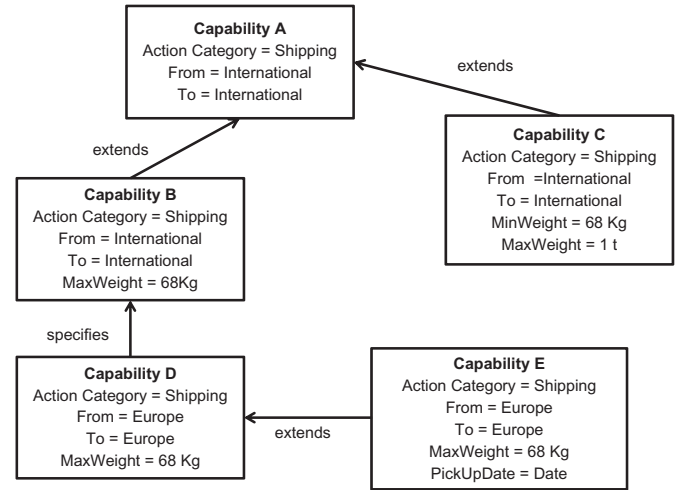


FIGURE 3. Example of Shipping Capabilities Hierarchy using coarse-grained relations.

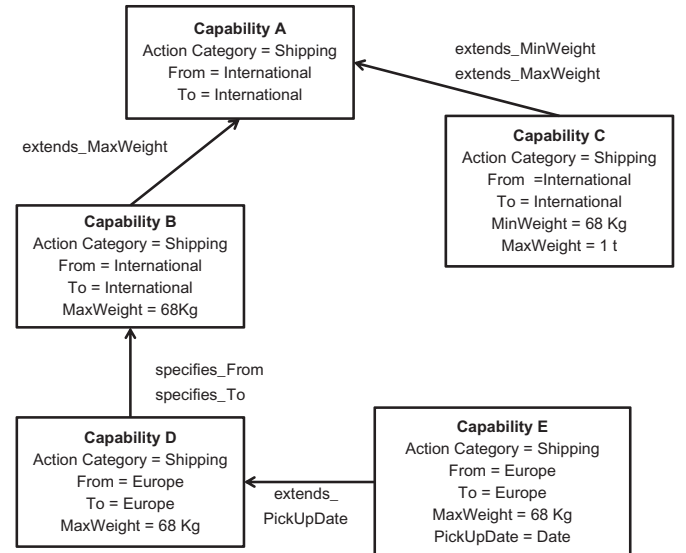


FIGURE 4. Example of Shipping Capabilities Hierarchy using fine-grained relations.

capabilities in a given domain can also be defined as an ontology where an agreement about their meaning is reached and shared. Like any other ontology concepts, these capabilities can be reused to define other ones. We implemented this model as a set of ontologies that are detailed in the following section.

4. IMPLEMENTING THE CAPABILITY META-MODEL AS RDF ONTOLOGIES

We have, recently, noticed a wide adoption of the Linked Data [32, 33] principles, and a growing amount of data sets

²Note that the notation for this example is not formal, formal descriptions of the used concepts is shown in Section 4.

specified in RDF. It is clear that Linked Data provide the best practices for publishing structured data on the Web. Linked Data are published using RDF where URIs are the means for referring various entities on the Web giving the possibility to interlink them. Currently, organizations are highly interested in publishing their data in RDF [34] as well as various public vocabularies (ontologies) are being released. Consequently, we have chosen to implement the proposed capability meta-model in RDF and make use of Linked Data principles in order to define capabilities, categories and properties as well as their values.

As illustrated in Fig. 5, we distinguish four levels of ontologies for implementing the proposed capability model:

- *Meta-Model*: is the lowest level that defines the required classes and properties for defining action categories (see Section 4.1) and property declarations (see Section 4.2).
- *Categories and Properties*: this level defines the set of categories (see Section 4.1) and properties (see Section 4.3) related to particular domain.
- *Domain Ontology*: is the actual capability domain ontology. It creates abstract capabilities that associate for each action category the possible set of properties (see Section 4.3).
- *Capabilities*: at this level, capabilities are created with respect to the capability domain ontology.

4.1. Action Categories

The action category meta-model proposes the set of classes and properties for defining the actions being performed in a particular domain. Figure 1 did not give any details about the *ActionCategory* class as it can be defined based on the implementation choices.

After analysing existing actions categories such NAICS [35], UNSPSC [36] and MIT Process Handbook [37], we propose to model action categories and relations between them (i.e. Meronymy and generalization relations). Meronymy (part-of) relations between action categories are used in NAICS [35] and MIT Process Handbook [37] and also investigated in [38] in order to capture granularity relations between actions at several levels of abstraction. The generalization relation is also used in NAICS [35] to represent that an action is more specific or general than another one (e.g. ‘book accommodation’ is more general than ‘book hotel’).

Figure 6 illustrates the proposed meta-model for action categories. We use in this ontology the prefix *ac* for the namespace <http://vocab.deri.ie/ac>.³ This ontology has only one *rdfs:Class* (rdf:type) and five *rdfs:Property* (rdf:type).

³Note that we also use the prefix *xsd* for the namespace <http://www.w3.org/2001/XMLSchema#>

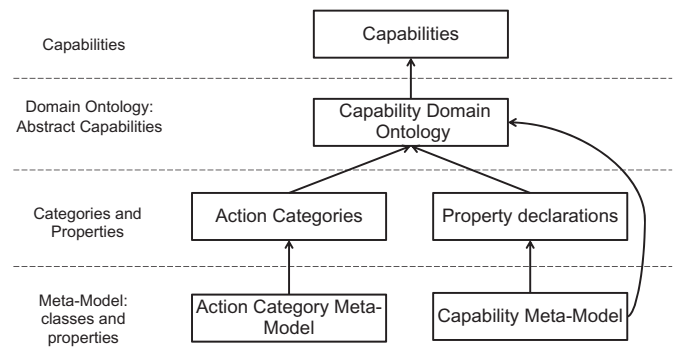


FIGURE 5. From meta-model to actual capabilities.

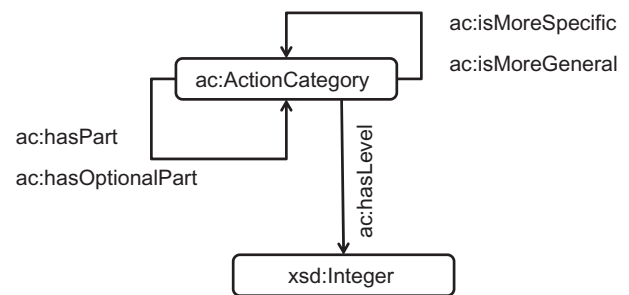


FIGURE 6. Action category meta-model.

ac:ActionCategory is the class of action categories. The properties *ac:hasPart* and *ac:hasOptionalPart* are used to create a meronymy hierarchy of action categories. An action is performed only if all its parts are performed. We use the property *ac:hasLevel* to assign the level (*xsd:Integer*) of an action category within a meronymy hierarchy of action categories. The top of the hierarchy can start from 0. The properties *ac:isMoreGeneral* and *ac:isMoreSpecific* (inverse relations) are also used to create a specification hierarchy of action categories.

This meta-model is used to create **domain-specific action categories** (called in this paper ‘*action categories ontology*’) and explicitly capture composition and generalization relations between them. To illustrate how it can be used, we translated an example of actions categories ontology from the MIT Process Handbook [37] of the *Distribute via Electronic Store* domain [39]. The resulting ontology is shown in Fig. 7 and Listing 1 shows the action category *deso:distributeBooksViaElectronicStore* using N3 [40] representation. The listing shows that the Action Category is at level 1 of the hierarchy of categories, this category is composed of three other ones; two are required: *deso:buyBooksToStoreAndToOrder* and *deso:sellViaElectronicStore*, and one is optional: *deso:manageSolelyInternetDistribution*.

4.2. Capability Meta-Model

Listing 2 shows the concept *cmm:Capability* as an *rdfs:Class* and *cmm:PropertyValue* as an equivalent class to *owl:*

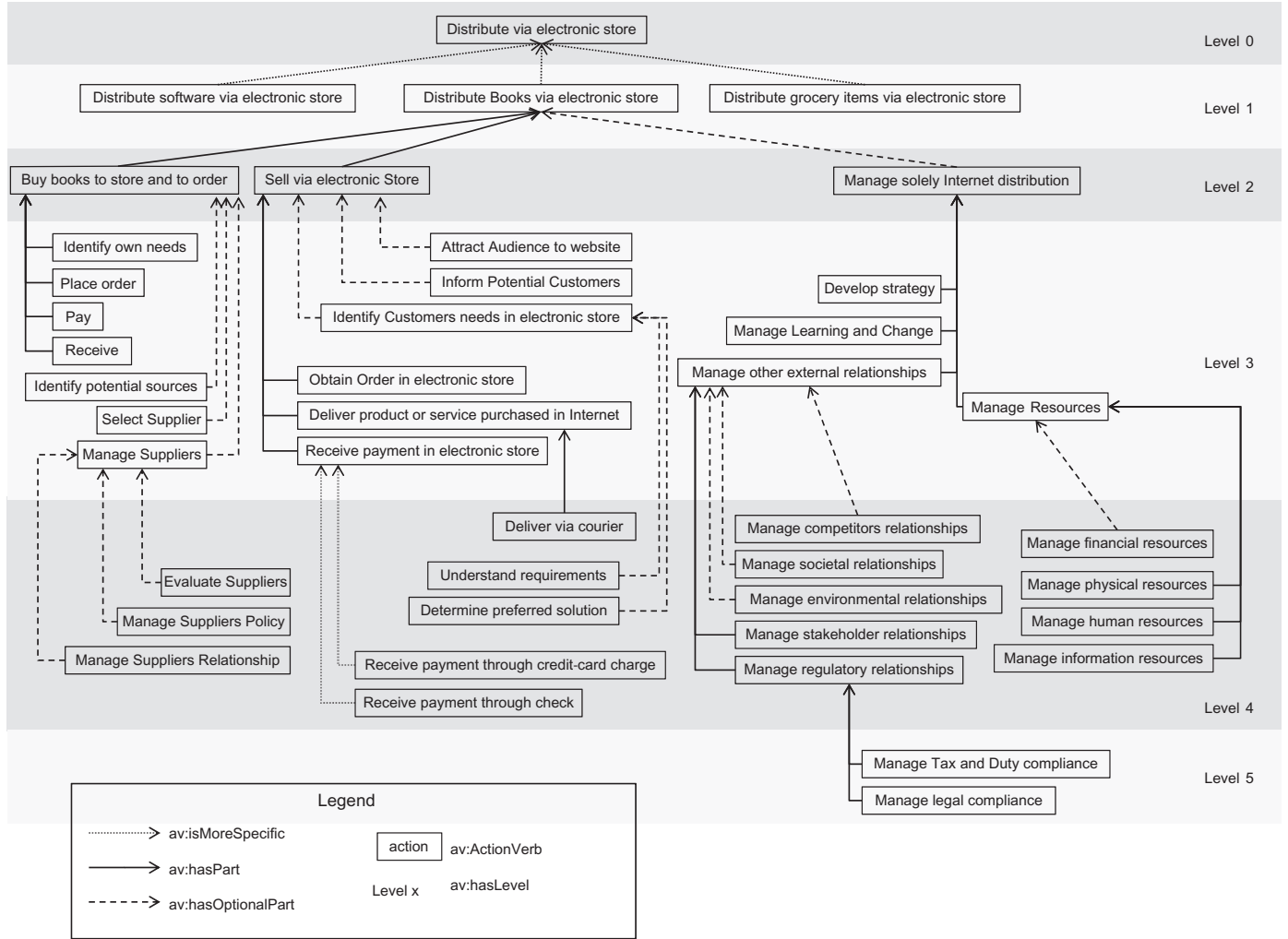


FIGURE 7. Example of action categories from the Distribute via Electronic Store Domain [39].

LISTING 1. Action categories ontology snippet from the distribute via Electronic Store Domain.

```

1 @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
2 @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
3 @prefix ac: <http://vocab.derri.ie/ac#>.
4 @prefix deso: <http://.../AContology/businessTravel#>.
5
6 deso:distributeBooksViaElectronicStore
7   a ac:ActionCategory;
8   ac:hasLevel '1'xsd:Integer;
9   ac:hasPart deso:buyBooksToStoreAndToOrder,
10             deso:sellViaElectronicStore;
11   ac:hasOptionalPart
12     deso:manageSolelyInternetDistribution;
13   rdfs:label 'Distribute books via electronic store'
14     xsd:String.

```

Thing because it needs to be the most general class to allow for the reuse of existing vocabularies for possible attribute values for example using *vcard* open vocabulary for defining addresses. Then, the property *comm:achieves* allows linking a *comm:Capability* to its corresponding *ac:ActionCategory*. Furthermore, the property *comm:property* allows to create

domain-specific properties that will be created as an *rdfs:subProperty* of *comm:property* and will be interpreted as properties of a capability. Finally, the property *comm:hasMostGeneralValue* is used to define the property declaration its most general value.

Note that Listing 2 is not a complete listing of the Capability Meta-Model, further details can be found in the online version.⁴

4.3. Property Declarations and Capability Domain Ontology

One can define a domain-specific capability ontology by modelling its action category and properties. We discuss in this section the various property types that our model

⁴Online version of *comm* available at: <http://vocab.derri.ie/comm> (accessed June 6, 2015).

LISTING 2. Capability meta-model snippet.

```

1 @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
2 @prefix owl: <http://www.w3.org/2002/07/owl#>.
3 @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
4 @prefix cmm: <http://vocab.deri.ie/cmm#>.
5 cmm:Capability a rdfs:Class.
6
7 cmm:PropertyValue owl:equivalentClass owl:Thing.
8
9 cmm:achieves a rdf:Property;
10   rdfs:domain cmm:Capability;
11   rdfs:range ac:ActionCategory.
12
13 cmm:property a rdf:Property;
14   rdfs:domain cmm:Capability;
15   rdfs:range cmm:PropertyValue.
16
17 cmm:hasMostGeneralValue a rdf:Property;
18   rdfs:domain cmm:property;
19   rdfs:range cmm:PropertyValue.
20

```

LISTING 3. Shipping domain ontology snippet.

```

1 @prefix vcard: <http://www.w3.org/2006/vcard/ns#>.
2 @prefix cmm: <http://vocab.deri.ie/cmm#>.
3 @prefix desco: <http://.../Dontology/businessTravel#>.
4
5 desco:deliverViaCourier a cmm:Capability;
6   cmm:achieves desco:deliverViaCourier;
7   desco:from vcard:VCard;
8   desco:to vcard:VCard.
9
10 desco:from rdfs:subProperty cmm:property;
11   rdfs:range vcard:VCard.
12
13 desco:to rdfs:subProperty cmm:property;
14   rdfs:range vcard:VCard.

```

supports. We will use the Distribute via Electronic Store domain (its set of action categories has already been discussed in the previous section) for defining the property declarations that are related to the action categories shown in Fig. 7.

Listing 3 shows the N3 [40] description of the capability *desco:deliverViaCourier* (see Line 5). Line 6 links this capability to its action verb from the Action Categories of the domain which is *desco:deliverViaCourier*. The rest of the listing illustrates how two properties *desco:from* and *desco:to* are defined. Both properties are *rdfs:subProperty* of *cmm:property* (Lines 10 and 13).

The range of the properties *desco:from* and *desco:to* is *vcard:VCard* to refer to an address. More complex and detailed property types are required for modelling more advanced properties. As depicted in Fig. 2, we consider five classes used for describing the values of a capability properties.

Using these classes separately or in combination, a capability can specify (i) the possible values properties can have, and (ii) how to compute their values. Before detailing these classes, we need to introduce the concepts of *Constraint* and *Expression* which some attribute values may refer to.

4.3.1. Constraint and Expression

A constraint enables to specify the possible values an attribute can have. The class *Constraint* represents all constraints. The

LISTING 4. Example of a constraint.

```

1 desco:PkgConstraint a cmm:Constraint;
2   cmm:hasExpression desco:PckConstraintExpr.
3
4 desco:PckConstraintExpr a cmm:Expression;
5   cmm:exprType "SPARQL";
6   cmm:exprValue "?weight <= 50? && ?weightUnit = dbpedia:KG".
7

```

LISTING 5. Example of a constrained value.

```

1 desco:deliverViaCourier desco:Item :X.
2
3 :X a desco:Package, desco:ConstrainedValue;
4   desco:hasWeight [desco:hasValue ?weight;
5   ship:hasUnit ?weightUnit];
6   cmm:constrainedBy desco:PckConstraint.
7

```

class *Expression* enables to specify expressions among them the value of a given constraint. The class *Expression* has two attributes/properties, *ExprType* which specifies the type of expression and *ExprValue* which defines the expression itself. The type of the expression, *ExprType*, indicates how to build the corresponding queries during a matching process. Currently, the only type of expression our meta-model supports is SPARQL (queries).

Listing 4 shows an example for expressing a constraint on the weight of the package. The constraint *PackConstraint* is defined in Line 1. This constraint has an expression of type SPARQL. The value of the constraint expression (Line 5) indicates that the weight of the package has to be lower than or equal to 50 kg.

4.3.2. Constrained Value

The class *ConstrainedValue* enables defining the possible values a property can have by specifying a set of constraints on its value. As depicted in Fig. 2, a *ConstrainedValue* is constrained by a set of constraints. Listing 5 shows how *desco:deliverViaCourier* can specify that it can deliver packages of weight under 50 kg. The value X, of the property *Item*, is a *ConstrainedValue* (Lines 1 and 3). X is constrained by the constraint *PckConstraint* (Line 6) which was detailed in Listing 4.

4.3.3. Dynamic Value

A *DynamicValue* defines how to compute the value of a property which value depends on (i) consumer provided properties, (ii) dynamic values or (iii) hidden variables. As shown in Fig. 2, a *DynamicValue* refers to an expression that defines how to compute it.

Listing 6 shows an example of how to compute the shipping price. The value Y, of the property *price*, is a *DynamicValue*. It has as evaluator the expression *PriceExpression* (Line 6) which is a SPARQL expression (Line 9). Line 10 specifies the formula for computing the price based on the weight of the package.

LISTING 6. Example of a dynamic value.

```

1  desco:deliverViaCourrier  desco:price  :Y.
2
3  :Y  a  desco:ShippingPrice , cmm:DynamicValue;
4      desco:hasValue  ?price;
5      desco:hasUnit  dbpedia:USD;
6      cmm:hasEvaluator  desco:PriceExpression .
7
8  desco:PriceExpression  a  cmm:Expression;
9      cmm:hasType  "SPARQL";
10     cmm:exprValue  "?price := fn:ceiling(?weight)
        *5.5+41".

```

4.3.4. Conditional Value

A *ConditionalValue* assigns a value to the corresponding property if a certain condition holds. That is, the value assignment itself is conditional. As shown in Fig. 2, a *ConditionalValue* has a condition expressed as a constraint and an element which corresponds to the corresponding property value.

Listing 7 gives an example showing how to specify a shipping price when the target country is a European country. The value *Y*, of the property *price*, is a *ConditionalValue* (Line 3). It assigns the Property Value, *EuropeanPrice*, when the *PriceCondition* holds (Line 5). *PriceCondition* is a *Constraint* which requires that the target country is in Europe (Lines 9–12). *EuropeanPrice* is a *DynamicValue* (Line 14) and has as evaluation expression *PriceExpression* which was detailed in Listing 6.

4.4. Updates from the Previous Version of the Model

As it has been previously mentioned, the model proposed in this paper is an update of previous efforts [30, 41]. The main updates in the concepts introduced in this paper are shown in Table 3.⁵ Other updates have also been proposed in other works for the modelling of sensor capabilities [31, 42] and for the modelling configurable process tasks [43].

4.5. Requirements' Satisfaction of the Business Capability Model

In this section, we discuss how the proposed model satisfies the units of analysis identified in Section 2. This evaluation methodology has also been carried out by *Oaks* for the evaluation of the proposed business capability meta-model in her thesis [44]:

- *Expressiveness—Explicitly represent the action performed*: the actions performed by a business capability are captured via the *rdfs* property **cmm: achieves**. The action categories are defined in an ontology of actions with composition and specification relations between them. This model can be further enriched by exploring other relations such as *synonymy*.

⁵Note the namespaces used: ac: <http://vocab.deri.ie/ac> cmm: <http://vocab.deri.ie/cmm> cap: <http://vocab.deri.ie/cap>

LISTING 7. Example of a conditional value.

```

1  desco:deliverViaCourrier  desco:price  :Y.
2
3  :Y  a  desco:ShippingPrice , cmm:ConditionalValue;
4      desco:hasValue  ?price;
5      cmm:hasCondition  desco:PriceCondition;
6      cmm:hasElement  :EuropeanPrice .
7
8
9  desco:PriceCondition  a  cmm:Constraint;
10     cmm:hasExpression  [cmm:hasType  "SPARQL";
11                        cmm:hasValue  "?trgCountry
12                        skos:subject  dbpedia-cat:
13                        European_countries"].
14
15  desco:EuropeanPrice  a  cmm:DynamicValue;
16     cmm:hasEvaluator  desco:PriceExpression .

```

- *Expressiveness—Explicitly capturing functional and non-functional features related to the action performed*: contrary to the IOPE paradigm, our model expresses a business capability with a set of properties that can be both functional and non-functional. The related properties of a particular high-level business capability are captured also in a domain-specific ontology.
- *Expressiveness—Ability to express these features using simple as well as complex types*: a property value of a business capability can be assigned any simple value such as string, integer or an address vcard (see Listing 1st:SDOnto). Furthermore, the proposed business capability meta-model proposes a set of advanced types such as *conditionalValue* and *enumerationValue* (see Fig. 2).
- *Inferences—Ability to explicitly identify relationships between business capabilities based on their descriptions*: the proposed model identifies two relations to capture specification (see Definition 3) and extension (see Definition 4) relations between business capabilities.
- *Inferences—Ability to index and search capabilities*: Example 2 *Hierarchy of Capabilities* shows how an indexing structure of business capability can be constructed based on their proprieties. However, this paper did not elaborate of how specification and extension relations can be extracted and used for building such structure. Future works will use this feature to build an indexing structure of business capabilities based on their properties.
- *Use of Ontologies—Use of domain or general ontological concepts for describing business capabilities*: actual business capabilities are derived from high-level ones that are defined in domain-specific ontologies. In the examples shown in this paper, we illustrated the use of general and domain-specific ontological concepts.
- *Use of Ontologies—Searching capabilities should not be relying on keyword extraction and comparison*: capability are described using ontological concepts, this allows the development of mechanisms of discovery using those concepts without relying on extraction of keywords from textual descriptions. Future works will focus on the indexing and discovery of business capability using this modelling approach.

TABLE 3. Mapping between the previous and the current version of the capability RDF concepts.

Concepts from our previous work [30, 41]	Concepts from this paper	Update Notes
cap:Capability cap:ActionVerb	cmm:Capability ac:ActionCategory	Equivalent rdf classes First, we replaced the cap:ActionVerb by a ac:ActionCategory as the term Verb was confusing. A category is not necessary a single verb as it was initially defined [41]. Second, cap:ActionVerb is a subclass of skos:Concept while ac:ActionVerb is an rdf:Class that introduces new properties: ac:hasLevel, ac:hasPart and ac:hasOptionalPart, etc. that are used to create a hierarchy of action categories
	ac:hasLevel ac:hasPart ac:OptionalPart ac:isMoreGeneral ac:isMoreSpecific	Introduced to create a hierarchy of action categories
cap:attribute	cmm:property	The difference between both concepts comes from the introduction of property declarations (see Definition 1)
cap:AttributeValue cap:variantOf	cmm:PropertyValue	Equivalent rdf classes Concept has been dropped from the current version
	cmm:hasMostGeneralValue	Introduced in the current version to infer specification relations between capabilities
cap:specifies	cmm:specifies	Similar properties with a further adjustment of their formal definitions
cap:extends cap:EnumerationValue cap:RangeValue cap:ConditionalValue cap:ConstrainedValue cap:DynamicValue cap:Constraint cap:Expression cap:hasExpression cap:hasElement cap:constrainedBy cap:hasEvaluator cap:hasCondition cap:hasMin cap:hasMax	cmm:extends cmm:EnumerationValue cmm:RangeValue cmm:ConditionalValue cmm:ConstrainedValue cmm:DynamicValue cmm:Constraint cmm:Expression cmm:hasExpression cmm:hasElement cmm:constrainedBy cmm:hasEvaluator cmm:hasCondition cmm:hasMin cmm:hasMax	Equivalent classes and properties

4.6. Annotating Process Models with Business Capabilities

In this section, we provide a proof of concept to show how we can use the capability model to annotate tasks of a process model.

It is important to note that our vision for describing business capabilities is to abstract from any service or process modelling language. In other words, a capability of a process task can be described in RDF outside the scope of the process model; while a link between the task and its capability needs to exist. Existing annotation mechanisms such as in Business Process modelling

Notation⁶ (BPMN for short) can be used to create this link but not for describing the capability. Using BPMN annotations as capability attributes loses the connection between the capability as an entity to its action categories and related properties. For other languages that do not provide annotation mechanisms such as EPC, additional changes in their specifications might be required.

As it is difficult for new users to write RDF annotations for describing the capability of their services or business processes,

⁶<http://www.bpmn.org/> (accessed October 11, 2017).

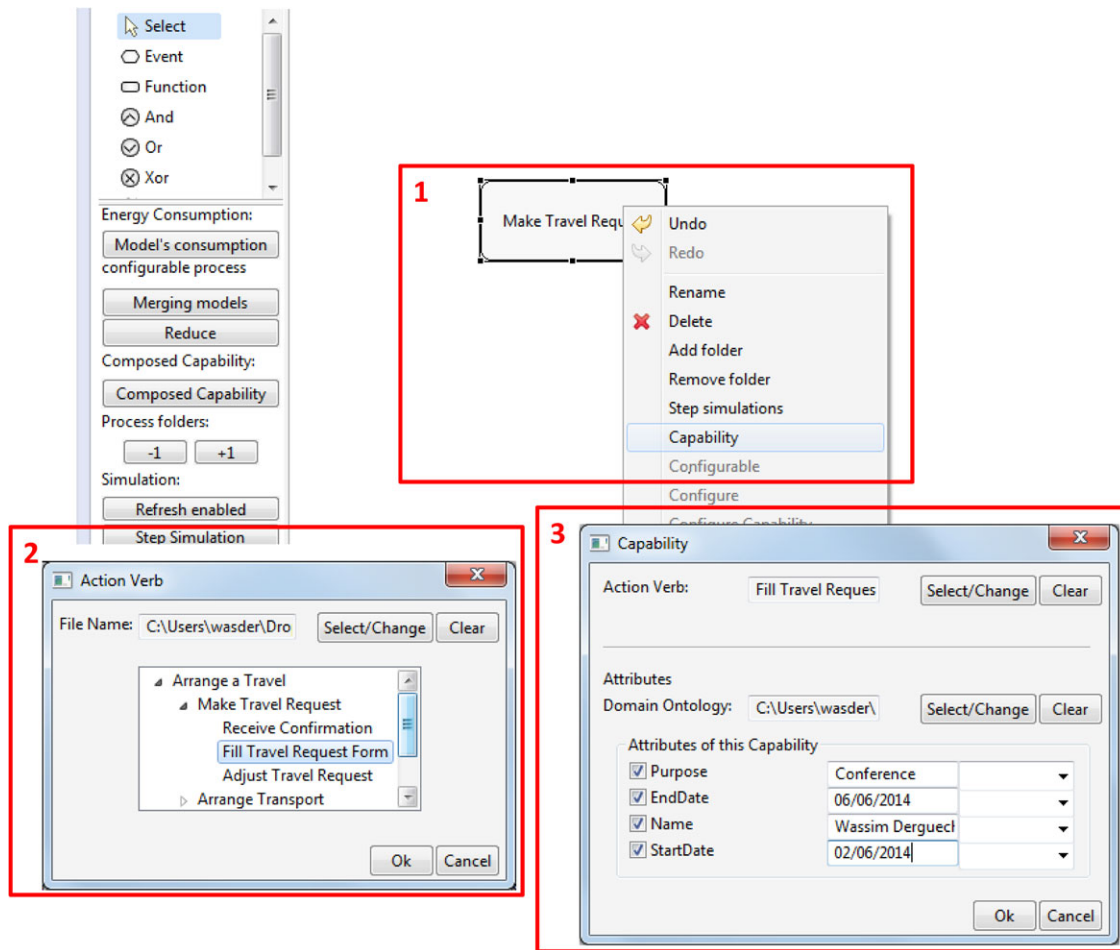


FIGURE 8. Extended version of EPCTools that supports annotation of business process tasks.

we have implemented an extension of EPCTools⁷ [45], a business process modelling tool using EPC, in order to assist users in the annotation operation. Figure 8 shows screenshots of the extended version of EPCTools. When a user wants to annotate a particular task of a model, he needs to right click it and choose *Capability* from the contextual menu (see 1 in Fig. 8). After loading the required ontologies, the user has to select the action category for the current task (see 2 in Fig. 8). Finally, the user has to choose what properties associated to the action category he wants to define in his capability (see 3 in Fig. 8).

The extended version of EPCTools offers two options to save the model either in its existing EMPL serialization with additional tags for the capability or exporting the entire model as RDF. Note that the primary purpose of this tool support is to provide a proof of concept for testing the applicability of the proposed model. A proper evaluation of the model itself is carried out in Section 6.

⁷The original version of EPCTools: <http://www2.cs.uni-paderborn.de/cs/kindler/Forschung/EPCTools/> (accessed November 30, 2015) and the new version is available at <https://goo.gl/iUcQNA>

5. CAPABILITY OF A BUSINESS PROCESS

While process models explicitly capture the involved activities and workflows together with organization-specific resources, the proposed capability model focuses at providing an abstract representation of what these processes achieve or the outcome that customers or collaborators need. Within the same organisation, there may be several workflows for specific outcomes (e.g. by rearranging activities and resources), but on a broader scale the organisation would not expose the different workflows, but would only show the business capabilities if offers. The model proposed can serve this need, however, at least one further steps is required: identification of the business capability of an entire process given that all of its activities are annotated with their business capabilities.

As illustrated in Fig. 9, determining the capability of an entire business process consists of determining its *ActionCategory* and associated *Properties*. Values of these concepts depend on the control flow of the process model as well as the capabilities of all its activities (i.e. cap1, cap2, cap3 and cap4 in Fig. 9).

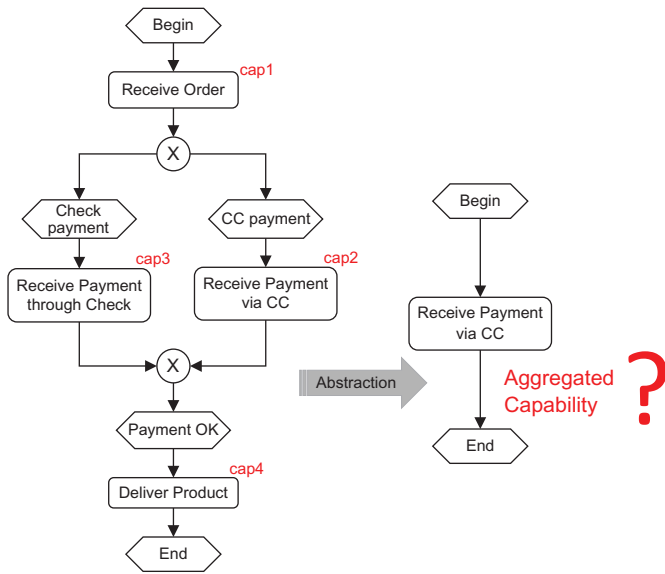


FIGURE 9. The capability of a business process is the aggregation of the capabilities of its activities.

5.1. Determining the Action Category

The action category is a mandatory property in the business capability description. Its value is taken from an Actions Ontology that is also used for determining the action category of aggregated business capabilities. An aggregated business capability has as action category corresponding to the Lowest Common Ancestor (LCA) of the action verbs of its components.

Using the ontology of Action Categories depicted in Fig. 7, consists of looking for the LCA of all the action verbs of the activities of that process model: $LCA(Obtain\ Order\ in\ electronic\ store, Receive\ payment\ through\ check, Receive\ Payment\ through\ Credit-card\ charge, Deliver\ Product\ of\ service\ purchased\ in\ Internet) = Sell\ via\ electronic\ Store$.

Ideally, all the action categories used in the model are taken from the same actions ontology like in this simple example. For various reasons, modellers can use actions taken from different action ontologies. Instead of searching for the LCA in a single actions ontology, one needs to take into account all the possible ontologies used in assigning action categories to the capabilities of a process model. In such a case, a more elaborated method is needed [38].

5.2. Determining the Properties

Properties of a capability of a business process model can be determined by propagating them from a start node to an end node. Each node introduces new properties with respect to various conditions (e.g. control flow such conditional branching). In order to propose a correct properties propagation algorithm, we assume that the input process model does not have any loops and is well structure. In a well-structured process model,

every split connector has a corresponding join connector, whereas both connectors bound a process model fragment with one entry node and one exit node [46]. We propose to use the formal semantics of the business capability annotated business process graph as a token game, similar to Petri Nets [47].

A Petri Net is a tuple (P, T, F) , where P is a finite set of places (representing events in EPC), T is a finite set of transitions ($P \cap T = \emptyset$) (representing functions in EPC) and $F \subseteq (P \times T) \cup (T \times P)$ is a set of arcs (flow relations). Petri nets can be used to represent dynamics of business process models by using token propagation to verify if models are regular, sound and well structured [48].

A token is a theoretical concept that is used as an aid to define the behaviour of a process by firing its nodes. The *Initial* place generates a token that traverses the sequence transitions and passes through all the places until reaching the *Final* place [49]. In this case, a process model is mapped to a petri net using transformation rules depending on the type of its nodes. In EPC, function and event nodes are transformed into transitions and places, respectively, while mapping connectors is more complex. This depends on the type of connector and its linked nodes [48].

The idea of propagating the properties of a process model is similar; it starts from the *InitialNode* then fires all the nodes one by one and propagates the subsequent properties until it reaches the *FinalNode*. Each node introduces some changes on the set of propagated properties. The propagated properties at a particular node are marked on its outgoing arcs.

The propagation of properties and conditions is then guided by the traversal of tokens in the petri nets representing the business process model. However, classical petri nets allow only the modelling of states, events, synchronizations, etc. and are not able to model data objects such as the properties of capabilities. To solve this issue, coloured or typed petri nets [50] have been introduced as an extension to classical petri nets where tokens represent objects (e.g. data item) in the system. Tokens represent 'colours' or set of properties. At each transition, a token is produced with respect to the consumed tokens. More concretely, a transition represents a relation between input and output tokens. Our proposed propagation algorithm [51] defines the relations of these transitions. The idea of this propagation has been used in the literature for propagating IOPEs of process models to determine their IT capabilities [52] as well as for the verification the soundness of business process models [53].

5.3. EPCTools Extension Implementing the Capability Aggregation Algorithm

The proposed business capabilities aggregation algorithm has also been implemented as an extended version of EPCTools⁸

⁸The original version of EPCTools: <http://www2.cs.uni-paderborn.de/cs/kindler/Forschung/EPCTools/> (accessed November 11, 2015) and the new version is available at <https://goo.gl/iUcQNA>.

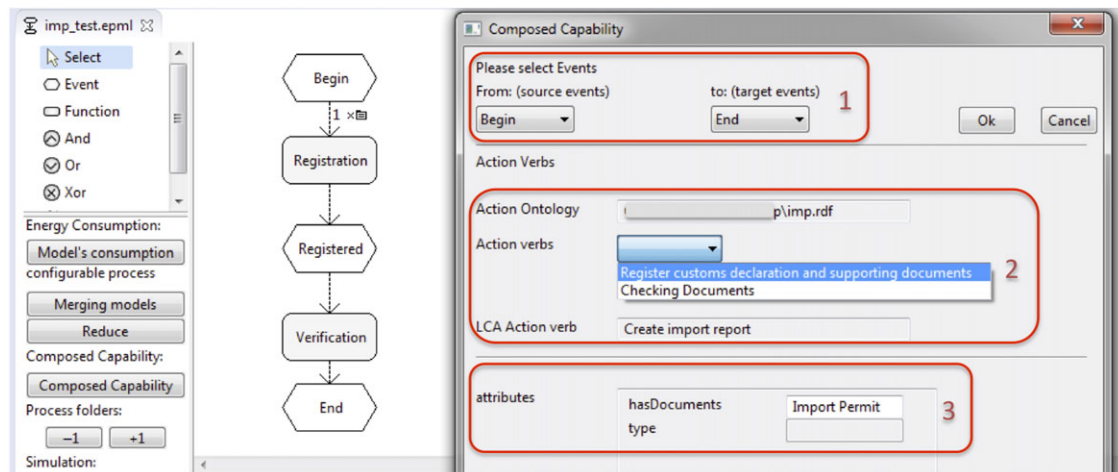


FIGURE 10. EPCTools extension implementing the capability aggregation algorithm.

[45]. This extension shown in Fig. 10 computes the business capability of a business process fragment defined by a start and end event (see Area 1 in Fig. 10). The result is shown to the user as a set of action categories (see Area 2 in Fig. 10) and a set of properties (see Area 3 in Fig. 10).

Note that the objective of this tool is to provide a proof of concept for determining the capability of an entire business process model. It has been developed to show the applicability of this approach, to carry out some manual verification of the results and to be used as a visual support for conducting interviews with domain experts. Apart from this running example, we manually tested this algorithm on a set of process models from the customs clearance processes: import procedures. The test collection includes 10 business processes that have been previously subject to a case study [54] for capability modelling in logistics. They describe guidance on the basic regulatory requirements that all importers must consider for importing goods. These processes involve various steps from submission of import documents until the release of the imported goods. The models were manually annotated using the Import Capabilities Domain Ontology (IMPC⁹). In Section 6.2, we report on the interviews carried out with domain experts.

6. EVALUATION OF THE META-MODEL

We propose to validate the business capability model proposed in this paper using ontological evaluation as a non-empirical evaluation method (see Section 6.1) and semi-structured interviews with domain experts as an empirical method (see Section 6.2).

6.1. Ontological Evaluation of the Business Capability Model

6.1.1. Introduction

The ontological evaluation of conceptual models consists of mapping the proposed conceptual model constructs to ontological concepts/constructs in order to assess the ability of the model to represent reality [12]. However, mapping the modelling language constructs to ontological concepts can be subjective especially when we want to identify intrinsic and non-intrinsic attributes or classes and kinds, consequently this can lead to a subjective evaluation. To avoid this issue, Wand *et al.* [11] propose to use a set of generic conceptual modelling constructs (i.e. instance, class and attribute) instead of the ontological constructs defined by Bunge [55]. In this approach, the evaluation of the model is carried out through the verification of a set of rules insuring that a model does not generate any semantic ambiguity by avoiding construct overload and redundancy. These rules are not related to a particular conceptual model and can be applicable to any model that is using generic ontological concepts. Therefore, we evaluated the proposed business capability conceptual model using this methodology.

6.1.2. Evaluation Steps

The first step of the evaluation consists of mapping the business capability conceptual model constructs to the generic conceptual model constructs proposed by Wand *et al.* [11] (i.e. instance, class and attribute). Table 4 shows the mapping that is used for the second step of this evaluation that consists of verifying that the model respects a set of rules defined by Wand *et al.* [11].

EVALUATION RULE 1. ‘Things are represented only as instances. Instances should represent only things.’ [11]

⁹<http://vocab.der.i.e/impcc>

In the business capability modelling approach proposed in this work as well as in other modelling paradigms (e.g. Object-Oriented modelling), *things* represent objects representing instances of classes. Rule 1 implies that *things* cannot be instances of attributes or other instances. This is true for all the attributes shown in Table 4 except for *Property Entry*. In Wand's constructs, there is no direct representation of properties. The model that includes such constructs should define how it can be interpreted. In this work, for creating a domain-specific business capability property, the model suggests to create an instance of a *Property Entry* that links a *Capability* instance to a certain *Value*. For this reason, in the implementation of this business capability meta-model, the *Property Entry* is defined as an *rdf:Property* (see *cmm:property* in Listing 2) and all business capability properties are created as *rdfs:subProperty* of *cmm:property* (see *desco:from* in Listing 3).

EVALUATION RULE 2. ‘Both simple and composite things should be represented using the same construct (entity, object).’ [11]

The business capability meta-model proposed in this paper does not differentiate between simple and composite business capabilities. Both can be presented as instances of the class *Capability*.

EVALUATION RULE 3. ‘A class or a kind of thing is defined in terms of a given set of attributes and relationships; that is, intrinsic attributes and mutual attributes.’ [11]

Instances of all classes in the business capability meta-model are defined by a set of attributes that are derived from intrinsic as well as mutual properties of the proposed model.

EVALUATION RULE 4. ‘An aggregate type/class must have properties in addition to those of its component types/classes.’ [11]

This rule does not apply in the case of the business capability meta-model proposed here as there are no aggregated classes in the model.

EVALUATION RULE 5. ‘All attributes and relationships in a class represent properties of things in the class.’ [11]

Instances of the classes proposed in the business capability meta-model are created as objects with a set of properties. These properties are either intrinsic or derived from mutual relation. Additional instances of business capabilities have additional attributes that are derived from the *Property Entry*.

EVALUATION RULE 6. ‘Null attributes have no meaning.’ [11]

The business capability meta-model does not allow the creation of instances without attributes. Most importantly, it

TABLE 4. Mapping the business capability conceptual model constructs to generic conceptual modelling constructs and ontological constructs.

Business capability conceptual model construct	Generic conceptual modelling construct [11]	Ontological construct [55]
Capability	Class	Class
Action Category	Class	Class
achieves	Attribute	Connection Attribute
specifies	Attribute	Attribute representing a mutual property
extends	Attribute	Attribute representing mutual property
Property Entry	No direct representation	Property
Property Name	Attribute	Attribute representing intrinsic property
Property Declaration	Class	Class
SpecRelation	Attribute	Attribute representing intrinsic property
Value	Class	Class
Range Value	Class	Class
hasMin	Attribute	Attribute representing intrinsic property
hasMax	Attribute	Attribute representing intrinsic property

does not allow the creation of a business capability without at least specifying its *Action Category* (i.e. an attribute generated from a mutual property).

EVALUATION RULE 7. ‘The same construct should be used to represent a binary relationship and a higher-order relationship.’ [11]

This rule is already covered by the mapping proposed by Wand *et al.* Indeed, both binary and higher-order relationships are mapped to Attributes. Furthermore, the business capability meta-model proposed in this paper does not have any higher-order relationships.

6.1.3. Discussion and Threads to Validity

The above-mentioned rules (i.e. Rules 1–7), as articulated by Wand *et al.* [11], verify that a general conceptual model can be used for modelling reality by avoiding construct overload and redundancy; which is the case with the business capability meta-model proposed in this paper.

Researchers such as Bunge [55], Wand *et al.* [11] and Weber [56] evaluate conceptual models from a realistic point of view. In other words, they assume that conceptual models should be designed to represent things that ‘exist in the

world'. The proposed business capability meta-model is, effectively, representing real things, i.e. actual actions performed by services, processes or human agents. The main observation is that business capabilities are not tangible objects and might need more flexibility to give the designer the possibility to model these actions as he perceives them. Weber [56] is in favour of such flexibility and argues that ontological foundations for information systems design might be subject to the perception of the designer. This is also aligned with paradigmatic approaches where conceptual models can capture different aspects of things depending on the intended perception and use of the conceptual models.

The idea of validating conceptual models using ontological analysis can be challenging and difficult to apply with complex models, especially when using the ontological analysis of Bunge [55]. This can be simplified by using the method proposed by Wand *et al.* [11] to avoid situations where the designer needs to differentiate between a 'kind' and a 'class'. The evaluation of the proposed business capability meta-model using this method was relatively simple as the model defines 14 constructs. This made the verification of the rules of Wand *et al.* [11] straightforward.

Even though in this work, we showed that the proposed business capability meta-model is 'suitable to represent reality', few threats of validity, as presented in Table 5, need to be considered for further assessments of the generated business capabilities to ensure that (1) they are consistent, (2) have an intuitive appeal to the end users, (3) they can be used to represent various domains and (4) can be used in empirical evaluations.

6.2. Interviews with Domain Experts

6.2.1. Introduction

In this part of the evaluation, we carry out two rounds of semi-structured interviews [13, 14] with 10 domain experts

that have strong background and are currently active in the area of service computing and information system design and development.

The objective of the two rounds is slightly different. The first one aims to assess the intuitive appeal of the capability modelling approach while the second examines the capability aggregation work. The reasons for these two interviews come from the fact that both contributions were not evaluated at the same time. First we conducted the assessment of the model, once validated we proceeded with the capability aggregation work and carried out the second round of interviews.

For each round of interviews, five different experts were invited. The interviews were done after explanation of the objective of this work and details about service modelling approaches. The main targeted outcome of these interviews was to identify if these experts can confirm that the proposed model is good enough to model business capabilities and if it can be adopted in their working environment.

6.2.2. Participants

For these semi-structured interviews, 10 participants were recruited from different levels of expertise in the area of service computing and information systems design and development. This number is estimated to be sufficient given that experts are not always available to take part of such experiments ([57] used seven experts in an experimental application of the Delphi method). The age group of these participants is 30–50 years old and their professional background includes a minimum of 5 years experience and are currently active in their field. The profiles of the experts interviewed in the first round include:

- two project managers (P1 and P2): leading teams of developers of information systems for the management of seaports in different countries;
- two service providers and consumers (P3 and P4): working as consultants in the area of telecommunication. One

TABLE 5. Threats to validity.

Threat	Description	Control technique
Consistency of business capabilities	Generated Capabilities should be consistent and compliant to the original model	RDF validators can be implemented to avoid creating capabilities with missing or wrong properties
Intuitive appeal of business capabilities	Generated capabilities can model irrelevant properties that not intuitive when describing the action of a service or a process	Validation with domain experts is required before using the model
Application to various domains of application	The meta-model should be applicable to various domains	The model allows the creation of new capabilities through new action categories and associated properties. This requires ontology engineering knowledge or a visual tool that generates capabilities
Use of the model in empirical evaluations	Using generated capabilities in empirical evaluations such as indexing, composition, discovery, etc.	Real or simulated data can be generated and used for empirical evaluations. We have experimented such evaluations for the indexing and discovery of sensor services [31]

of them is also a manager of his own start-up offering automated post services;

- and one IT engineer (P5): working in the same start-up as the main developer of the provided service.

The profiles of the experts interviewed in the second round include

- two system architects (P6 and P7): working as designers of information systems for clients of a multi-national company;
- one project manager (P8): leading teams of developers of information systems for the management of seaports in different countries;
- one IT engineer (P9): another developer from the start-up offering automated post services;
- and one information system consultant (P10): working as a consultant and trainer in the area of business process management.

6.2.3. Approach

The approach used for both rounds of interviews follows the case study research process proposed in [58]:

First round of interviews:

- *Case study design*: the objective of the first round of interviews is to assess the intuitive appeal of the proposed model and the objective for the second is to assess the usefulness and intuitive appeal of the proposed aggregation algorithm for identifying the business capabilities of a process model. Interviews run individually using online conferencing and desktop sharing tools (to allow the experts to use the tool themselves remotely). Each interview took about 1 h for each participant.
- *Preparation for data collection*: The discussions were semi-structured to give the participants the freedom to give additional comments and get as much feedback as possible from them.
- *Collecting evidence*: The structure of the first round of interviews was as follows¹⁰:

- (1) 5 min discussion about the profile of the participant and his knowledge about services and business processes modelling languages.
- (2) 15 min presentation of the business capability meta-model introduced in this work with open discussion on each component and its use.
- (3) 15 min for creating simple business capability ontology in RDF
- (4) 10 min demo and interaction with the tool support

- (5) 15 min discussion about the proposed approach and modelling language

And the structure of the second round of interviews was as follows:

- (1) 5 min discussion about the profile of the participant and his knowledge about services and business processes modelling languages.
- (2) 15 min presentation of the business capability of atomic and aggregated tasks and the propagation algorithm introduced in this chapter with open discussion.
- (3) 15 min for manually defining the aggregated capability of a simple process model.
- (4) 10 min demo and interaction with the tool support.
- (5) 15 min discussion about the proposed business capability aggregation approach.

- *Analysis of collected data*: A post interview analysis of the collected feedback is reported in Section 6.2.4.
- *Reporting*: A discussion of the resulting feedback is summarized in Section 6.2.5 and shared among the participants.

6.2.4. Results

The following outcomes were identified:

General comments on the experience of the experts in capability modelling

- The Experts P1–P5 are familiar with all the standardized service description languages that will be discussed in Section 7: WSMO [22], OWL-S [23], SAWSDL [24, 25] and SA-REST [26].
- Each of these experts (P1–P5) has extensively worked with at least one of these languages.
- It is highly agreed by P1, P2, P3 and P4 that all these languages focus primarily on the technical aspect without considering the business aspect.
- None of these experts (P1–P5) is still using any of these languages because they are complicated to understand and get familiar with.
- Developers need to read a lot about the use of ontologies, rules, reasoners, etc. An average user cannot easily adopt such technologies.
- P1–P5 prefer to have their own custom made modelling of their assets.
- These experts (P1–P5) are developing RESTful APIs that exchange data using JSON format and for them using JSON for their services description was a natural choice.
- The use of JSON gives them the flexibility they need to identify their own properties and values.

¹⁰Note that the durations used here are approximative. Some of the interviews run for few minutes more or less for each section.

Feedback on the business capability modelling approach

- These experts confirmed that modelling of business capabilities the way we propose in this paper is close to their vision and simple to implement.
'Frame-based modelling is same as key value pairs. This is exactly how we model objects in JSON.' (P5).
- The Experts P6–P10 find that the business capabilities modelling is a promising direction towards end-users understanding. *Actions* are what users do in their processes and properties use business terms that these experts are familiar with. Two of the experts highlight that the main advantages that the model brings is the simplicity and extensibility.
'It looks like your model can also capture other aspects!' (P6)
'You are proposing a simpler view of what the model achieves.' (P10).
- Designing sample capabilities by P2 and P5 was easy and intuitive after a short tutorial.
- None of the experts is in favour of the idea of exclusively modelling business capability properties.
- They are interested in including more implementation/technical properties.
- The Experts P6–P10 find that the adoption of this work into their information systems is possible as long as long as it can be adapted to their modelling and annotation techniques.

Feedback on the aggregation of capabilities (identification of the capability of an entire business process)

- Regarding the aggregation work, the Participants P6–P9 find it as a useful feature for users that are developing multiple service-based business processes. It allows not only identifying the business capability of the model but also any other aspect of interest and visualize the parameters used or required by the process.
'I can add here another property regarding the data format used in one activity to make sure that it is communicated to the following activities as a requirement.' (P9).
- The consultant and training expert (P10) finds that the results of aggregation can be further used for documentation purposes. This can help delivering business processes and services documentation quickly.
- P6, P8 and P9 see that abstraction and aggregation techniques are already in use in their working environment. However, they do not use rich descriptions of models and services and thus do not have rich abstracted service or process descriptions.

Feedback on the implementation of the business capability meta-model

- Ontologies and taxonomies are already in use and constitute valuable assets for the companies where these experts work. All the experts use ontologies.
- The experts (P2, P3 and P4) were not necessarily in favour of using RDF as an underlying implementation language of such model. After discussing JSON-LD, they were convinced that it is a better alternative.
'I think RDF is not the best implementation language.' (P2).

Feedback on the tool support

- The tool support was very useful to show how the model can be used to annotate business process models.
- While testing the aggregation tool support, the Experts P6–P10 find that it is simple to use and intuitive. It is important to note that in the implemented prototype we used only primitive types.
- Experts (P6–P10) do not see using simple types as a major issue as they find that properties are more important to visualize than their values.
'At this stage I don't care about the values used, I give more importance to the parameters themselves!' (P7).

6.2.5. Summary of the Interviews' Outcomes

The important outcomes from these interviews can be summarized as follows:

- All the expert agree that current modelling languages do not give much importance to business capabilities. The proposed model in this work comes as an addition rather than a substitution to current models for describing a different view of enterprise information systems.
- Frame-based modelling is a good modelling paradigm towards ease of use and intuitiveness of the models.
- The tool support was very helpful to hide the complexity of RDF as an underlying realization language.
- Some of the experts suggested to include technical aspects in the current model to serve as bridge between both business and IT perspectives. This recommendation is aligned with our vision in this work. The business capability is one of the aspects of a service description that can be further extended.
- A rich capability can then easily be transformed into a complete documentation using Natural Language Generation techniques [59, 60].

6.2.6. Discussion and Threads to Validity

Surveys and interviews are often characterized with a high degree of representativeness compared to experiments [61]. However, they exhibit a low level of control over extraneous factors such as the influence of the background of the participants to their answers. To limit this factor, in this evaluation,

the choice of these particular participants was made for two main reasons. First of all, the managers P1 and P8 were involved in various informal discussions about the proposed model. Consequently, they are familiar with this research and particularly with the motivation of modelling business capabilities. Second, the diversity in these profiles guarantees different points of view:

- Managers have a global view over the entire lifecycle of business processes.
- Consultants have multiple interactions with end-users as they, regularly, give trainings and information sessions to end-users.
- Engineers have a strong technical background in the development of services and business process management tools.

7. RELATED WORK

A capability denotes what an action does either in terms of world effects or returned information [18]. The purpose of providing well-defined capabilities of services or business processes is to allow end-users to discover them with respect to the action they perform. In this section, we are particularly interested in service description languages and how they describe capabilities of services. We classify these contributions in three families: Semantic Web Services models, Semantic Annotation of Invocation Interfaces models and Frame-based models. These three families are discussed in details in the following three sections.

7.1. Semantic Web Services models

The first family of contributions for capability descriptions includes Semantic Web Services models (WSMO [22, 62] and OWL-S [23, 63]). Capability descriptions with this family are split into information transformation and state of the world change captured as Input, Output, Preconditions and Effects (IOPE paradigm) [64]. A recent contribution that comes to resolve issues of rigidity service description using WSMO [22] and OWL-S [23] is Simple Semantic Web Architecture and Protocol (SSWAP) [65]. It provides yet another paradigm for service descriptions that uses description logics. It has the particularity to add further service meta-data such as provider details and taxonomic descriptions of data items [66].

Critique: Both OWL-S and WSMO were designed at a time when extensive service descriptions were thought to be effective to build a web service architecture. The major problem with these languages is that they have been extensively enriched making the description of the entire service in some cases complex. Furthermore, describing what a service would do upon the change of state of the world after its execution

proved to be a much harder problem than developers of WSMO and OWL-S anticipated. This created a strong dependency on service descriptions that SSWAP proposed a solution to resolve it.

In these solutions, information transformation and state of the world changes define the capability of a service expressed in terms of axioms, consequently the explicit action performed is not captured. However, in OWL-S Profiles, a classification in a service taxonomy such as NAICS [35] or UNSPSC [36] can be used to help identify the actual action being performed and in SSWAP a textual description of the service can be used as such.

7.2. Semantic Annotation of Invocation Interfaces Models

The second family of related efforts concerns semantic annotations of invocation interfaces (SA-WSDL [24, 25] and SA-REST [26, 67]). While these approaches do not directly target capability modelling, they attempt to provide alternative solutions to top-down semantic approaches (WSMO [22, 62] and OWL-S [23, 63]) by starting from existing descriptions such as WSDL [19] and annotate them with semantic information.

Critique: These contributions focused mainly on annotating the service interfaces rather than functional *capabilities*. This is mainly perceived in the fact that there were no clear decisions regarding the attributes to be used in their specifications. The researchers that worked on these contributions could have taken the decision to use RDFS/OWL model that defines terms like ‘category’, ‘precondition’, ‘effect’, etc. that would allow service descriptions to be typed in a standard way.

7.3. Frame-based models

The third family includes frame-based approaches for modelling capabilities. This is another way to describe capabilities featuring *functional declarations* that are different from the classical IOPEs. Functional declarations are investigated in details by researchers from the linguistics and Natural Language Processing (NLP) domain with the aim to give another view on the structure of sentences by describing verbs using ‘cases’ contained in case frames [68]. Example of cases include agent (who), location (where) and instrument (how) as declared by *Fillmore*.

The idea of modelling capabilities using frames has been used to describe the capabilities of software agents [28] and proves to be effective for enhancing agents’ communication and planning while facilitating human understanding of agents capabilities. Celino *et al.* [69] use the same approach for describing data and services published on the web. In the same vision, Oaks *et al.* used frame-based modelling for describing service capabilities. Oaks *et al.* proposed a comprehensive conceptual model that extends IOPEs paradigm

with additional frames extracted from textual descriptions. Frames used by Oaks *et al.* are similar to those defined by Fillmore. It makes the model easy for humans to read and understand but machines won't be able to use this model to compose capabilities. Composition is still relying on the classical IOPEs.

7.4. Summary and discussion

This section reviewed related approaches that proposed service description models that can be used as alternatives/extensions to either simple labels and textual descriptions or to existing languages such as WSDL [19]. A summary of these approaches is in Table 6.

All the proposed approaches are reliable for carrying out machine processing operations such as composition and discovery. These solutions were proposed to avoid relying on simple labels or long textual descriptions in these operations. However, most of the proposed approaches do not go beyond the classical IOPEs. This makes search requests exclusively defining the state of the world before and after the execution of a service, something that has been proven to be difficult [70, 71] and requires additional abstraction efforts to make end-users able to query services in a more user friendly manner [70, 71].

The most important highlights that we noticed while analyzing these approaches are:

- *Explicit actions* even using simple lexical terms form a good basis for a capability description. This is a natural way human users define what a service or application does. Capturing these actions in a domain-specific ontology helps improve their reuse and creating a common understanding on their semantics.
- Capability descriptions models should be *open* to allow for more *flexibility* to end-users to include their own ontological concepts and their own way to describe their assets. A good example is the quick and high adoption of JSON as a simple format for exchanging and modelling structured data without strict restrictions on what attributes to use nor any particular order that they should follow, etc.
- Enriching the action performed with explicit *functional and non-functional properties* does not only refine further the action being carried out but also can be used to infer relations between capabilities. These relations can create an indexing structure that is not exclusively built on a categorization schema of lexical terms.

8. CONCLUSION

Process aware information systems' stakeholders range from the IT department engineers that are responsible for the

development and monitoring of IT activities within and organization to the domain experts that are responsible for its growth, innovation and response to market demands. A primary requirement for developing an information system that serves the needs of all these stakeholders, is to integrate in its processing in addition to the control-flow perspective (i.e. what activities are involved in a process and how they are ordered, etc.) the other relevant perspectives: i.e. data flow perspective: how data is flowing in between the different process activities, organizational perspective: what are roles are required for each activity, and the functional perspective: what capabilities are achieved by each process element (i.e. activity, process fragment or entire process).

In this work, we have been particularly interested in the integration of the functional perspective into service descriptions and business process models. In our state-of-the-art analysis, we found that even though a proper capability description is recognized to be important for automated discovery, composition, indexing, etc. of services and business process models, little effort has been put towards describing this concept as standalone entity. It has always been part of other concepts such as services and invocation interfaces or simply reduced a simple label or textual description.

In our work, we consider a capability as standalone entity that can exist outside the scope of service descriptions or invocation interfaces. A service, a computer programme, a business process or even a manual task can be described using this concept where an explicit link can be created between them. In a very simple definition, we consider a capability as a set of actions enriched with zero or many properties. Properties allow to refine further the action that is taken for a domain-related ontology. For example, shipping services can be described using the action category 'shipping' that can be extended with properties reporting on the 'source address', 'destination address', etc. One can argue that such properties are also present in current service description. This is true but these are rather part of the input and output parameters of the services rather than its capability that is described via the state change of the world before and after the execution of a service (i.e. precondition and effect).

The modelling of capabilities as a set of actions and properties is inspired by the frame-based modelling approaches that have been proven to be effective in practice with languages such as JSON. It is simple, relies mainly on a shared agreements on the semantics of the used actions and properties that are defined common ontologies. The other advantage of using frame-based modelling is the possibility of indexing, searching and aggregating capabilities without heavily relying on reasoning which is the major issue with current approaches. Detailed analysis of current modelling approaches was carried out in Section 7 and the details of the capability meta-model are discussed in Section 3. Details about the implementation of the conceptual model (i.e. capability meta-model) are discussed in Section 4 and its evaluation is reported in Section 6.

TABLE 6. Comparative analysis of capability modelling approaches.

	Expressiveness			Inferences		Use of Ontologies	
	Requirement 1.1: Action Performed	Requirement 1.2: Related functional and non-functional features	Requirement 1.3: Complex and Simple Types	Requirement 2.1: Relations between capabilities	Requirement 2.2: Index and Search	Requirement 3.1: Use of Ontological concepts	Requirement 3.2: Not relying on keyword extraction search
Semantic Web Services Models: WSMO [22] and OWL-S [23]	Partially fulfilled as in OWL-S Profiles one can use categories of services using taxonomies such as NAICS [35] or UNSPSC [36] (but remains not explicit action description)	Partially fulfilled as IOPEs do not feature in an explicit and easily accessible way domain features. Additional effort towards the extraction of these features is required e.g. [72]	Fulfilled as both WSMO and OWL-S have extensions to allow for describing complex types	Partially fulfilled as both languages claim to have support for creating relations between services, but I could not find any work that used this feature. Both languages are proposed to enable automation of discovery	Fulfilled as the profile of a service in OWL-S can be optionally positioned in a hierarchy of profiles	Fulfilled as both languages use domain or general ontological concepts	Fulfilled
Semantic Annotation of Invocation Interfaces Models: SA-WSDL [24, 25] and SA-REST [26]	Partially fulfilled as the <i>modelReference</i> in SA-WSDL can optionally be used for categorization [5]	Not fulfilled as these approaches describe interaction interfaces rather than concrete capabilities	Fulfilled as semantic annotations allow for describing complex types	Not fulfilled as relations between interfaces descriptions are used to determine potential interactions that can be used for composition	Partially fulfilled as the <i>modelReference</i> can be used for categorization that can be used for indexing but this remains extremely limited	Fulfilled as both languages use domain or general ontological concepts	Fulfilled
Frame-based Models: Oaks <i>et al.</i>	Fulfilled as the model proposed by Oaks <i>et al.</i> distinguished the action verb of the capability	Not fulfilled as the model simply adds to the classical IOPE an action verb and <i>informational attributes</i> that are neither explicitly capturing functional and non-functional features nor capturing domain-related properties	Fulfilled as the proposed model is rich enough to model both simple and complex types	Partially fulfilled as the model proposes to use relations between action verbs in terms of synonymy, equivalence, etc. But this has not been validated/ tested	Partially fulfilled as the model uses many informal attributes that can be used for classification/ indexing. However, the authors do not elaborate further on this requirement	Partially fulfilled as the model allows using domain or general ontological concepts however more efforts are put towards using lexical-based terms	Fulfilled

As a future research direction, it is possible to investigate the use of NLP techniques for providing suggestions of business capabilities. This can be used either to fully automate the annotations of services and processes with their capabilities or propose autocompletions during their manual annotations. For example, a business capability can be linked to the so called

‘case frames’ in linguistics as it is offered by FrameNet project [73]. In linguistic study, the capability of an action is defined by an *action verb* that is quite similar to the *action category* in our proposed model. Then several dimensions may extend that verb by giving more details about the carried work and related aspects. While in FrameNet these dimensions are predefined

such as agentive, dative, and objective in our case, we do not impose any predefined dimensions. However, it is possible to establish links between both models to generate structured business capabilities from textual descriptions using linguistic case frames. Such idea has been discussed in the literature [74–76] for automatically annotating process activities with their action verbs derived from the textual documentation; or exploring the idea of mapping case frame dimensions to capability properties to generate a full structured capability using the model proposed in this paper [77].

Furthermore, the business capability as implemented in this work is well structured. It explicitly lists for its properties various types of values: conditional, range, enumeration, etc. Each of these types has a clear definition on its semantics. In this regard, Natural Language Generation (NLG) techniques can be used to provide a textual description of each of these values. This can be extended to the entire capability description and generate a textual description that serves as a documentation of services and processes. This can also explore another dimension of human understanding by providing customized summary of the capability using natural language.

FUNDING

This publication received the financial support of Science Foundation Ireland (SFI) under Grant Number SFI/12/RC/2289.

REFERENCES

- [1] Mendling, J., Reijers, H.A. and Recker, J. (2010) Activity labeling in process modeling: empirical insights and recommendations. *Inf. Syst.*, **35**, 467–482.
- [2] Mendling, J., Recker, J. and Reijers, H.A. (2010) On the usage of labels and icons in business process modeling. *IJISMD*, **1**, 40–58.
- [3] Hepp, M. and Wechselberger, A. (2008) OntonaviERP: Ontology-Supported Navigation in ERP Software Documentation. In Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T.W. and Thirunarayan, K. (eds.) *The Semantic Web—ISWC 2008*, 7th Int. Semant. Web Conf., ISWC 2008, Karlsruhe, Germany, October 26–30, 2008. *Proceedings, Lecture Notes in Computer Science*, **5318**, pp. 764–776. Springer.
- [4] Roman, D., de Bruijn, J., Mocan, A., Lausen, H., Domingue, J., Bussler, C. and Fensel, D. (2006) WWW: wsmo, wsml, and WSMX in a Nutshell. In Mizoguchi, R., Shi, Z. and Giunchiglia, F. (eds.) *The Semantic Web—ASWC 2006*, First Asian Semant. Web Conf., Beijing, China, September 3–7, 2006. *Proceedings, Lecture Notes in Computer Science*, **4185**, pp. 516–522. Springer.
- [5] Martin, D., Paolucci, M. and Wagner, M. (2007) Bringing Semantic Annotations to Web Services: OWL-S from the SAWSDL Perspective. In *The Semantic Web, 6th Int. Semant. Web Conf., 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007*, Busan, Korea, 11–15 November, Lecture Notes in Computer Science, **4825**, pp. 340–352. Springer, Berlin, Heidelberg.
- [6] Kopecký, J., Vitvar, T., Bournez, C. and Farrell, J. (2007) SAWSDL: semantic annotations for WSDL and XML schema. *IEEE Internet Comput.*, **11**, 60–67.
- [7] Lathem, J., Gomadam, K. and Sheth, A.P. (2007) SA-REST and (s)mashups: Adding Semantics to Restful Services. In *Proc. First IEEE Int. Conf. Semantic Computing (ICSC 2007)*, September 17–19, 2007, Irvine, California, USA, pp. 469–476. IEEE Computer Society.
- [8] Oaks, P., ter Hofstede, A.H.M. and Edmond, D. (2003) Capabilities: Describing What Services Can Do. In Orłowska, M. E., Weerawarana, S., Papazoglou, M.P. and Yang, J. (eds.) *ICSOC, Lecture Notes in Computer Science*, **2910**, pp. 1–16. Springer.
- [9] Roman, D., Kopecký, J., Vitvar, T., Domingue, J. and Fensel, D. (2015) WSMO-Lite and hRESTS: lightweight semantic annotations for Web services and RESTful APIs. *Web Semant.*, **31**, 39–58.
- [10] Berners-Lee, T., Hendler, J. and Lassila, O. (2001) The semantic web. *Sci. Am.*, **284**, 34–43.
- [11] Wand, Y., Storey, V.C. and Weber, R. (1999) An ontological analysis of the relationship construct in conceptual model. *ACM Trans. Database Syst.*, **24**, 494–528.
- [12] Yair, W. and Ron, W. (1989) An Ontological Evaluation of System Analysis and Design Methods. In Falkenberg, E. and Lindgreen, P. (eds.) *Information Systems Concepts: An In-depth Analysis*. Elsevier Science Publishers, Amsterdam, The Netherlands.
- [13] Bodart, F., Patel, A., Sim, M. and Weber, R. (2001) Should optional properties be used in conceptual modelling? A theory and three empirical tests. *Inf. Syst. Res.*, **12**, 384–405.
- [14] Drever, E., Scottish Council for Research in Education (1995) *Using Semi-structured Interviews in Small-Scale Research: A Teacher's Guide*. SCRE Publication. Scottish Council for Research in Education.
- [15] Dutta, S., Narasimhan, O. and Rajiv, S. (2005) Conceptualizing and measuring capabilities: methodology and empirical application. *Strategic Management Journal*, **26**, 277–285.
- [16] Amit, R. and Schoemaker, P.J.H. (1993) Strategic assets and organizational rent. *Strateg. Manage. J.*, **14**, 33–46.
- [17] Weske, M. (2007) *Business Process Management: Concepts, Languages, Architectures*. Springer.
- [18] OASIS (2006). OASIS reference model for service oriented architecture 1.0. <http://www.oasis-open.org/committees/download.php/19679/soa-rm-cs.pdf>. Accessed: 25/05/2014.
- [19] WSDL (2001). WSDL: Web Services Description Language. <http://www.w3.org/TR/wsdl>. Accessed: 25/05/2014.
- [20] Scheer, A.-W. and Schneider, K. (2006) ARIS—Architecture of Integrated Information Systems. In Bernus, P., Mertins, K. and Schmidt, G. (eds.) *Handbook on Architectures of Information Systems*. Springer, Berlin Heidelberg.
- [21] Sycara, K.P., Widoff, S., Klusch, M. and Lu, J. (2002) Larks: dynamic matchmaking among heterogeneous software agents in cyberspace. *Auton. Agent. Multi. Agent. Syst.*, **5**, 173–203.

- [22] WSMO (2005). WSMO: Web service modelling ontology. <http://www.wsmo.org/>. Accessed: 25/05/2014.
- [23] OWL-S (2004). OWL-S: Semantic markup for web services. <http://www.w3.org/Submission/OWL-S/>. Accessed: 25/05/2014.
- [24] SA-WSDL (2007). SA-WSDL: Semantic Annotations for WSDL. <http://www.w3.org/2002/ws/sawsdl/>. Accessed: 25/05/2014.
- [25] Verma, K. and Sheth, A.P. (2007) Semantically annotating a web service. *IEEE Internet Comput.*, **11**, 83–85.
- [26] SA-REST (2010). SA-REST: Semantic annotation of web resources. <http://www.w3.org/Submission/SA-REST/>. Accessed: 25/05/2014.
- [27] Wickler, G.J. (1999) Using Expressive and Flexible Action Representations to Reason about Capabilities for Intelligent Agent Cooperation. PhD thesis University of Edinburgh, Edinburgh, UK.
- [28] Wickler, G. and Tate, A. (1999) Capability representations for brokering: A survey. Available from: www.aiai.ed.ac.uk/project/oplan/cdl/cdl-ker.ps.
- [29] Gao, F. and Derguech, W. (2012) Ubiquitous Service Capability Modeling and Similarity Based Searching. In Haller, A., Huang, G., Huang, Z., Paik, H. and Sheng, Q.Z. (eds.) Web Information Systems Engineering—WISE 2011 and 2012 Workshops—Combined WISE 2011 and WISE 2012 Workshops, Revised Selected Papers, Sydney, Australia, 28–30 November, *Lecture Notes in Computer Science*, **7652**, pp. 173–184. Springer.
- [30] Derguech, W. and Bhiri, S. (2013) Modelling, Interlinking and Discovering Capabilities. In *ACS Int. Conf. Computer Systems and Applications, AICCSA 2013*, Ifrane, Morocco, May 27–30, pp. 1–8.
- [31] Derguech, W., Bhiri, S., Hasan, S. and Curry, E. (2015) Using formal concept analysis for organizing and discovering sensor capabilities. *Comput. J.*, **58**, 356–367.
- [32] Bizer, C., Heath, T. and Berners-Lee, T. (2009) Linked Data - The Story So Far. *IJWSIS*, **5**.
- [33] Sheridan, J. and Tennison, J. (2010) Linking UK Government Data. *LDOW 2010*.
- [34] Lebo, T. and Williams, G.T. (2010) Converting governmental datasets into linked data. *I-SEMANTICS*. ACM.
- [35] NAICS (1997) North American Industry Classification System (NAICS). <http://www.census.gov/eos/www/naics/> (accessed May 25, 2014).
- [36] UNSPSC (1998) United Nations Standard Products and Services Code (UNSPSC). <http://www.census.gov/eos/www/naics/> (accessed May 25, 2014).
- [37] MIT (2001) MIT process handbook. <http://process.mit.edu/> (accessed May 25, 2014).
- [38] Smirnov, S., Dijkman, R.M., Mendling, J. and Weske, M. (2010) Meronymy-Based Aggregation of Activities in Business Process Models. In Parsons, J., Saeki, M., Shoval, P., Woo, C. C. and Wand, Y. (eds.) Conceptual Modeling—ER 2010, 29th Int. Conf. Conceptual Modeling, Vancouver, BC, Canada, November 1–4, 2010. *Proceedings, Lecture Notes in Computer Science*, **6412**, pp. 1–14. Springer.
- [39] SAP (2001) Distribute via electronic store.
- [40] Berners-Lee, T. and Connolly, D. (2008) Notation3 (n3): A readable rdf syntax. W3c team submission. W3C.
- [41] Bhiri, S., Derguech, W. and Zaremba, M. (2012) Modelling Capabilities as Attribute-Featured Entities. In Cordeiro, J. and Krempels, K. (eds.) Web Information Systems and Technologies—8th Int. Conf., WEBIST 2012, Porto, Portugal, April 18–21, 2012, Revised Selected Papers, *Lecture Notes in Business Information Processing*, **140**, pp. 70–85. Springer.
- [42] Baccar, S., Derguech, W., Curry, E. and Abid, M. (2015) Modeling and Querying Sensor Services Using Ontologies. In Abramowicz, W. (ed.) Business Information Systems—18th Int. Conf., BIS 2015, Poznań, Poland, June 24–26, 2015, *Proceedings, Lecture Notes in Business Information Processing*, **208**, pp. 90–101. Springer.
- [43] Derguech, W., Bhiri, S. and Curry, E. (2017) Designing business capability-aware configurable process models. *Inf. Syst.*, **72**, 77–94.
- [44] Oaks, P. (2005) Enabling Ad-hoc Interaction with Electronic Services. PhD thesis Faculty of Information Technology, Queensland University of Technology Brisbane, Australia.
- [45] Nicolas, C. and Ekkart, K. (2006) EPC Tools.
- [46] Kiepuszewski, B., ter Hofstede, A.H.M. and Bussler, C. (2000) On Structured Workflow Modelling. In Wangler, B. and Bergman, L. (eds.) Advanced Information Systems Engineering, 12th Int. Conf. CAiSE 2000, Stockholm, Sweden, June 5–9, 2000, *Proceedings, Lecture Notes in Computer Science*, **1789**, pp. 431–445. Springer.
- [47] Peterson, J.L. (1981) *Petri Net Theory and the Modeling of Systems*. Prentice Hall PTR, Upper Saddle River, NJ, USA.
- [48] van der Aalst, W.M.P. (1999) Formalization and verification of event-driven process chains. *Inf. Softw. Technol.*, **41**, 639–650.
- [49] Istoan, P. (2012) Defining Composition Operators for BPMN. In Gschwind, T., Paoli, F.D., Gruhn, V. and Book, M. (eds.) Software Composition—11th Int. Conf., SC 2012, Prague, Czech Republic, May 31–June 1, 2012. *Proceedings, Lecture Notes in Computer Science*, **7306**, pp. 17–34. Springer.
- [50] van der Aalst, W. (1994) Putting high-level petri nets to work in industry. *Comput. Ind.*, **25**, 45–54.
- [51] Derguech, W. and Bhiri, S. (2011) Merging Business Process Variants. In Abramowicz, W. (ed.) Business Information Systems—4th Int. Conf., BIS 2011, Poznan, Poland, June 15–17, 2011. *Proceedings, Lecture Notes in Business Information Processing*, **87**, pp. 86–97. Springer.
- [52] Vulcu, G., Bhiri, S., Derguech, W. and Ibáñez, M.J. (2011) Semantically-enabled Business Process Models Discovery. *Int. J. Bus. Process Integration Manage.*, **5**, 257–272.
- [53] Weber, I., Hoffmann, J. and Mendling, J. (2010) Beyond soundness: on the verification of semantic business process models. *Distributed Parallel Databases*, **27**, 271–343.
- [54] Derguech, W. and Bhiri, S. (2012) Capability Modelling—Case of Logistics Capabilities. In Rosa, M.L. and Soffer, P. (eds.) Business Process Management Workshops—BPM 2012 International Workshops, Tallinn, Estonia, September 3, 2012. *Revised Papers, Lecture Notes in Business Information Processing*, **132**, pp. 519–529. Springer.

- [55] Bunge, M. (1977) *Treatise on Basic Philosophy. Ontology I: The Furniture of the World*. Riedel, Boston.
- [56] Weber, R. *et al* (1997) *Ontological Foundations of Information Systems*. Coopers & Lybrand and the Accounting Association of Australia, New Zealand, Melbourne.
- [57] Dalkey, N. and Helmer, O. (1963) An experimental application of the Delphi method to the use of experts. *Manage. Sci.*, **9**, 458–467.
- [58] Runeson, P. and Höst, M. (2009) Guidelines for conducting and reporting case study research in software engineering. *Empir. Softw. Eng.*, **14**, 131–164.
- [59] Leopold, H., Mendling, J. and Polyvyanyy, A. (2012) Generating Natural Language Texts from Business Process Models. In Ralyté, J., Franch, X., Brinkkemper, S. and Wrycza, S. (eds.) *Advanced Information Systems Engineering—24th Int. Conf., CAiSE 2012, Gdansk, Poland, June 25–29, 2012. Proceedings, Lecture Notes in Computer Science*, **7328**, pp. 64–79. Springer.
- [60] Leopold, H., Mendling, J., Reijers, H.A. and Rosa, M.L. (2014) Simplifying process model abstraction: techniques for generating model names. *Inf. Syst.*, **39**, 134–151.
- [61] Siau, K. and Rossi, M. (2011) Evaluation techniques for systems analysis and design modelling methods—a review and comparative analysis. *Inf. Syst. J.*, **21**, 249–268.
- [62] Roman, D., Kopecký, J., Vitvar, T., Domingue, J. and Fensel, D. (2015) Wsmo-lite and hrests: lightweight semantic annotations for web services and restful apis. *J. Web Sem.*, **31**, 39–58.
- [63] Sheng, B., Zhang, C., Yin, X., Lu, Q., Cheng, Y., Xiao, T. and Liu, H. (2016) Common intelligent semantic matching engines of cloud manufacturing service based on owl-s. *Int. J. Adv. Manuf. Technol.*, **84**, 103–118.
- [64] Roman, D., Keller, U., Lausen, H., de Bruijn, J., Lara, R., Stollberg, M., Polleres, A., Feier, C., Bussler, C. and Fensel, D. (2005) Web service modeling ontology. *Appl. Ontol.*, **1**, 77–106.
- [65] Gessler, D., Schiltz, G.S., May, G.D., Avraham, S., Town, C. D., Grant, D.M. and Nelson, R.T. (2009) SSWAP: a simple semantic web architecture and protocol for semantic web services. *BMC Bioinf.*, **10**, 309.
- [66] Nelson, R.T., Avraham, S., Shoemaker, R.C., May, G.D., Ware, D. and Gessler, D. (2010) Applications and methods utilizing the simple semantic web architecture and protocol (SSWAP) for bioinformatics resource discovery and disparate data and service integration. *BioData Min.*, **3**, 3.
- [67] Luo, C., Zheng, Z., Wu, X., Yang, F. and Zhao, Y. (2016) Automated structural semantic annotation for restful services. *IJWGS*, **12**, 26–41.
- [68] Fillmore, C.J. (1968) The Case for Case. In Bach, E. and Harms, R.T. (eds.) *Universals in Linguistic Theory*, pp.0–88. Holt, Rinehart and Winston, New York.
- [69] Celino, D.R., Reis, L.V., Martins, B.F. and Souza, V.E.S. (2016) A Framework-based Approach for the Integration of Web-based Information Systems on the Semantic Web. In *Proc. 22Nd Brazilian Symp. Multimedia and the Web*, New York, NY, USA. Webmedia '16, pp. 231–238. ACM.
- [70] Zaremba, M., Vitvar, T., Bhiri, S., Derguech, W. and Gao, F. (2012) Service Offer Descriptions and Expressive Search Requests—Key Enablers of Late Service Binding. In Huemer, C. and Lops, P. (eds.) *E-Commerce and Web Technologies—13th Int. Conf., EC-Web 2012, Vienna, Austria, September 4–5, 2012. Proceedings, Lecture Notes in Business Information Processing*, **123**, pp. 50–62. Springer.
- [71] Chouiref, Z., Belkhir, A., Benouaret, K. and Hadjali, A. (2016) A fuzzy framework for efficient user-centric web service selection. *Appl. Soft Comput.*, **41**, 51–65.
- [72] Kamaruddin, L.A., Shen, J. and Beydoun, G. (2012) Evaluating Usage of WSMO and OWL-S in Semantic Web Services. In Ghose, A. and Ferrarotti, F. (eds.) *Eighth Asia-Pacif. Conf. Conceptual Modelling, APCCM 2012, Melbourne, Australia, January 2012, CRPIT*, **130**, pp. 53–58. Australian Computer Society.
- [73] Baker, C.F., Fillmore, C.J. and Lowe, J.B. (1998) The Berkeley FrameNet project. In *COLING-ACL '98: Proceedings of the Conference*, pp. 86–90. Montreal, Canada.
- [74] Leopold, H., Smirnov, S. and Mendling, J. (2012) On the refactoring of activity labels in business process models. *Inf. Syst.*, **37**, 443–459.
- [75] Rahm, E. and Bernstein, P.A. (2001) A survey of approaches to automatic schema matching. *VLDB J.*, **10**, 334–350.
- [76] Mendling, J. and Simon, C. (2006) Business Process Design by View Integration. In *Business Process Management Workshops, BPM 2006 Int. Workshops, BPD, BPI, ENEI, GPWW, DPM, semantics4ws*, Vienna, Austria, September 4–7, 2006, *Proceedings, Lecture Notes in Computer Science*, **4103**, pp. 55–64. Springer.
- [77] Gao, F. and Bhiri, S. (2014) Capability Annotation of Actions Based on their Textual Descriptions. In Reddy, S. (ed.) *2014 IEEE 23rd Int. WETICE Conf., WETICE 2014, Parma, Italy, 23–25 June, 2014*, pp.257–262. IEEE Computer Society.