We are IntechOpen, the world's leading publisher of Open Access books
Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS
BOOK CITATION INDEX
INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

Chapter 5

E-Service Composition Ontology

Farzad Sanati

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.68467

Abstract

Recently, the concept of life event was coined in the literature to describe an event in a citizen's life that would require at least one or in some cases a collection of public services from e-government service composition prospective. A system to provide the required intelligence that would be able to compose the basic e-services to build a composite service has recently become an area of debate with many solutions being proposed. This study building on prior publications is attempting to introduce and provide new technical guidelines for a new ontology that extends Ontology Web Language for Services (OWL-S) as knowledgebase and intelligent actor for creating composite services to build a life event. This chapter proposes the life-event ontology that is a logical extension of OWL-S for implementation of e-service integration modelling framework proposed in prior publications.

Keywords: ontology, e-service composition, e-government, life event

1. Introduction

An intelligent software needs to provide a platform-independent description for services that it renders, while providing the means by which the service is accessed. Then a delivery platform is needed as such where descriptions of services are made and shared and delivered to clients also in an independent platform. The delivery platform should be able to employ a standard ontology that consists of a set of basic classes and properties for declaring and describing services. Ontology Web Language (OWL) seems to be the best candidate for this from the ontology structuring mechanisms point of view. OWL provides an appropriate representation language framework within which to do this.

This chapter provides formal methods for design and proof of an upper ontology, which is of crucial importance for e-service composition framework. It discusses the definition of 'ontology',

INTECH
open science | open minds

© 2018 The Author(s). Licensee InTech. This chapter is distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/3.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited. CC BY

'service ontology' and 'life-event ontology'. It gives details of what the concepts of service and service ontology mean in this study. This study explains and illustrates the overview of conceptual design for life-event ontology through the annotation used in the formal method using first-order logic to establish the main axioms and design rules for the 'life-event ontology' [1].

Ontology described in this chapter will provide technical support for implementation of life-event ontology-oriented service composition platform. Life-event ontology is a logical extension of Ontology Web Language for Services (OWL-S), extending this existing approach enables the design of a life-event metamodel, which in turn is used as an alterable workflow for composite services. The resulting ontology covers specific web services with semantic concepts to implement the conceptual life-event framework in the context of e-service composition by

1. Facilitating the construction of alternative integrated service workflows from entirely different web service vendors.
2. Enabling the repair or reconfiguration of life-event workflows in runtime.
3. The invocation of web services according to the workflow sequence.

2. Definition of ontology

In the domain of information technology, the term ontology is a word borrowed from philosophy that refers to the science of describing the kind of entities in the word and how they are related [2]. According to the Merriam Webster online dictionary, it is defined as[1]

1. A branch of metaphysics concerned with nature and the relations of beings.
2. A particular theory about the nature of being or the kind of things that have existence.

In the context of information technology, the most typical kind of ontology has a taxonomy and a set of inference rules [3]. On the other hand, a more formal definition of ontology is given by Maedche [4] as follows:

An ontology is a five-tuple as

$$O ::= (C, R, H^C, rel, A^O) \tag{1}$$

where C is a set of classes (concepts) and R is a set of relations.

Therefore, $H^C \subseteq C \times C$ is called taxonomy,

$H^C(c_1, c_2)$ means c_1 '*is-a*' c_2;

$rel : R \rightarrow C \times C$ is a function defined for other relations;

And, A^o is a logical language.

[1] http://www.merriam-webster.com/dictionary/ontology

In addition to these definitions, ontologies are used to describe a variety of models, ranging from simplest taxonomies such as Online Directory Project[2] to very complex knowledge models written in first-order logic.[3]

2.1. RDF and RDF schema

The resource description framework (RDF) is a recommendation of W3C specifications, originally designed as a metadata model. It is used as a general method for conceptual description or modelling of information that is implemented in web resources, using a variety of syntax formats.

RDF has an eXtensible Markup Language (XML)-based syntax, it provides a metadata model for ontology knowledgebase and also provides a common framework so that applications can process and exchange the information automatically through the World Wide Web.

RDF schema (RDFS) is the description of RDF language. It is also expressed in XML and provides a simple ontology of RDF concepts and property definitions. RDF provides the basis on which next generation ontology languages are developed. In other words, most new ontology languages are logical extensions of RDF.

2.2. RDF model

RDF identifies objects using Unified Resource Identification (URI) [5]. The literature describes a URI to be a series of characters that identifies an abstract or physical resource. URL is used to identify resources (available date) on the Internet, these resources be in form of html web pages or other form of data made available in the XML form. RDF uses simple property terms and their values to describe resources. RDF statements are represented as triples consisting of subject, predicate and object. The subject signifies the resource, the predicate represents the relationship between the subject and the object. The object can itself be a resource or a string literal, which represents a basic data type such as Integer or Boolean values.

RDF schema as the semantic extension of RDF describes and maps out all the relationships between resources in RDF. The resources are divided into groups or classes, which they provide a simple hierarchical classification structure, which relates these classes to each another through their properties. There is also 'Predicate' or property element, which represents the predicate and object of the statement. Its content is the object of the statement, which described as plain literal.

2.3. OWL

Recent advances have resulted in new developments in RDF towards a more cohesive approach towards new representation of knowledge. One of the most talked about is Ontology Web Language (OWL). OWL ontology is also an RDF graph and is represented by a set of RDF triples. As with any RDF graph, an OWL ontology graph has different syntactic forms. RDF

[2]http://www.dmoz.org/.
[3]http://www.ehealthserver.com/ontology/.

and RDF schema ontologies are extended by OWL via adding more terminology for describing properties and classes as well as cardinality, equality, relations between classes, richer property typing, property characteristics and enumerated classes.

Pure hierarchy of classes and their relationships is not the only thing that the ontology is defining; they are also used to inference class relationships such as equivalence or being disjoint.

2.4. OWL model

The root class of OWL ontology is called owlThing. Other classes defined within the ontology are subclasses of owlThing. OWL also supports a set of operators on classes such as union, intersection and complement. It also allows class enumeration and disjoint. There are two types of simple properties in owl description: one is 'data type' and the other one is 'object' properties. Relationship between class instances and RDF literals or XML schema data types is defined as data type properties. Relationship between the instances of classes is defined as object properties. There could also be logical connectivity such as transitivity, inverse and functional and symmetric connections. Similar to RDFS, an OWL class can contain instances of individuals of the class and other subclasses [6].

According to RDF, instances are defined to be the descriptions of a class with certain values in their properties. On the other hand, in OWL a class could also be defined with logical restrictions based on some properties. Classes can also be restricted by existential or universal quantifiers. For instance, the class life event may have subclasses defined with existential quantifiers on the hasService property such as 'hasService some Life-eventService' [1].

Hence, to restrict a subclass of the life-event class, one can define it as the MotorcycleLicenseLife-event class, which contains all life events that have Life-eventService as a service. This includes all the instances that have the hasService property assigned to the Life-eventService. In addition, these properties can also have cardinality restrictions. Therefore, one can safely say life event must have at least one Life-eventService but may have more than one. Thus, the hasService property can be restricted to

hasService > 1 and a life-event instance can have multiple hasService properties.

OWL has three sublanguages: OWL Full, OWL DL and OWL Lite, all described in Ref. [7]. OWL Lite is the least expressive of them. OWL Lite is somewhat more expressive than RDFS, because it provides simple constraints of classes and properties in addition to supporting a classification hierarchy. OWL DL is modelled on description logics, all conclusions are guaranteed to be computable that means, it supports maximum expressiveness while retaining computational completeness. A service class does not need to be an individual in order to represent a collection of individuals. This study suggests the use of OWL reasoners such as FaCT++ [8] to check for the accuracy of OWL documents.

2.5. OWL querying

OWL ontology can be queried in several ways. One effective way is to use a reasoner in order to create a class with certain constrains, then by classifying the class within the ontology, one can see which classes it relates to. All other relationships of this class such as equivalent classes,

http://dx.doi.org/10.5772/intechopen.68467

super classes and subclasses can also be inspected by the query processor. Reasoners can also be used to query over the OWL instances. The query processor in this method converts the query to an OWL class instance, where all the property values are the same as that of the query, where the reasoner can find all the classes that have this instance as an inferred instance. An inferred instance is known to be one that has not been explicitly instantiated within a class but is inferred to be part of a class as a result of the values and type of its properties. There are more OWL query languages that have been developed besides these methods. One example of OWL query language is the OWL-QL [9], which is a formal language and protocol that queries an OWL ontology by finding class relationships. It also allows querying and answering agents to conduct a query-answering dialogue in ontologies represented by OWL.

3. Ontology versus database

The reasoning power of ontology is the motivation behind its use for representing services rather than using simple attribute-value representations of data such as in traditional databases. An example of a query which can be done using an ontology which is difficult to do using a Simple Query Language (SQL) query is 'Given a service class, find all logically related matches to my query'. Simple Query Language (SQL) also does not support abstract data types, thus making it difficult to determine whether a certain property value belongs to a number of different classes or types. Ontologies can also be shared, re-used and changed. Ontologies can be distributed across the Internet and grow limitlessly, and they can be discovered and shared using their URI.

When new relationships are established within the ontology schema because of ontology migration or the addition of new classes, determining new relationships within the ontology is simply reduced to running a reasoner on the ontology in order to reclassify the classes. For relational databases, changes to the schema may have a fundamental impact on the existing data.

The main drawback to using ontology is that classification is expensive. As ontologies grow large, and especially when instances of classes are stored in the ontology, reasoning becomes a bottleneck. We tackle this problem by storing instances in separate ontology data-files instead of the ontology schema itself. This speeds up the classification process considerably. Moreover, a positive side effect of the distributed architecture of Life-event Ontology Oriented Service Integration (LOOSI) is that it allows each component to handle different ontologies (OWL-S and life event).

3.1. Web service ontology

In order to understand what real-world services are, we look at some works in the economic and business sciences as well as literature originating from Information and Communication Technology (ICT) area.

Hardly any of the generic concepts concerning real-world services show up in current e-commerce product classifications or standards for web services, a term that refers to Internet-based technologies, rather than business activities. Web services are loosely coupled reusable software components that semantically encapsulate discrete functionality. Web services,

however necessary and useful, cannot really be seen as services in the sense of the business science literature, they are currently rather restricted to input/output interface specifications.

There are many initiatives that have made progress towards defining and organising ontologies for web services; two of which have contributed a great deal to the state-of-the-art:

1. OWL-S describes a set of foundation classes and properties that can be used for declaring and describing services. One example of that is described in W3C 2004 documentation, stating that 'an ontology for describing Web Services that enable users and software agents to automatically discover, invoke, compose and monitor Web resources offering services under specified constraints' [10]. OWL-S tries to cover the description of services in a wide sense, not focusing on a particular application domain or problem.
2. Web Services Modelling Ontology (WSMO) intended to solve the composition problem by creating ontology for describing different aspects of semantic web services, but with a more defined focus. WSMO also takes into account the specific application domains (e-commerce and e-work) in order to ensure the applicability of the ontology for these areas.

In comparison of the two standards, WSMO includes majority of the elements present in OWL-S while adding new elements in order to grow its relevance in most domains. Mediation and compensation are examples of such characteristics that are key issues yet to be solved in order to achieve the real implementation of semantic web services, in such a way to be relevant for e-commerce [11]. A more thorough look at WSMO reviles that it should provide a higher level of detail for the definition of aspects such as choreography or grounding required by web service implementation. If these elements are appropriately covered, then WSMO can become a strict superset of OWL-S that also covers relevant issues not covered by OWL-S. WSMO also intends to have an execution platform, called web service modelling eXecution (WSMX) environment, while the intentions of OWL-S in this direction are not yet defined.

A web service transaction involves three parties: the service requesters, the service provider and a mediation infrastructure facility. The service requester, who may broadly be identified as a user, seeks a service to complete its task; the service provider, who can be broadly identified as a provider, provides a service sought by the user. The user may not know of the existence of the provider ahead of time, so it relies on mediator infrastructure facilities that act like a service registry and workflow organiser to find the appropriate provider. The role of the mediator registry is to match the request with the offers of service providers to identify which of them is the best match. In Chapters 6 and 7, we will provide the details of such facilities that can act as a delivery platform for life event and a framework for using such a platform. The remainder of the section explains the detailed design for an ontology that can provide a foundation for such a framework.

Consider the utilisation of a credit card service. A customer can choose the simplest form of a credit card or a more expensive card, which offers extra services as free travel insurance, high withdrawal limits, travel assistance abroad, worldwide card replacement in the case of loss, linking the card to a preferred supplier and many more. Multiple aspects of this service offering can be facilitated by websites: ordering a card, transactions listing, buying other services and goods with the card and more.

Another example is the online organisation of events such as conferences, board meetings, executive courses and exhibitions. Their electronic facilitation requires many capabilities, including a good predefined classification of such events, together with a description of their properties, plus the constraints they impose such as suitable times and spaces (rooms, halls and room set-up).

Essentially, an ontology is needed that can define the core contents of the service. In addition, electronic facilities should provide the capability of selecting relevant supplementary services. Such services include travel insurance and high withdrawal limits in the credit card case, and coffee breaks, video facilities, Internet connection, translation, sound, technical assistance or catering, in the event organisation case. This, once more can only occur in a predefined and standardised ontology-oriented way, in a way that related additional restrictions and relationships can be automatically provided for. The other issue is where service consumer requirements regarding a service usually contain some implicit things, fuzzy statements, and often requires a substantial interpretation. This steps into the service provider's ontological terminology and the components that the service provider can actually deliver.

OWL-S has made progress towards configuration-based service composition. This study suggests that an important element of a paradigm for the support of real-world services is a generic description of services and what they provide. Simply, a service ontology that the runtime design and production of services can be simplified to a configuration task. This task of looking up configuration in order to build composite services is called service composition. This can be translated in to a collaborative e-government scenario, where the ideal is to have an intelligent support system that [12]

- ontology has service bundle contents,
- interprets the preferences and customer needs into suitable terms from the perspective of the service provider, and
- can deal with all the associated restrictions while automatically constructing the requested service in a configuration-oriented way, which supports the composition of services from different service providers into a user-controlled workflow of composite web services.

The biggest limitation of OWL-S is that it only allows for the composition of services (or operations) described in one Web Services Definition Language (WSDL). A new system of web service knowledgebase configuration would be necessary if we wanted to compose web services from different vendors. One major challenge is that the service ontology must be sufficiently generic to be useful across many application domains. We discuss below how such an ontology might look and present its logical formal description.

4. Life-event ontology description

This research defines the life-event ontology as the logical extension of OWL-S to provide extended functionality, which allows a systematic composition of e-government web services

by means of abstraction in design and implementation. The advantage of such an extension is that it preserves and uses all the capabilities inherited from upper ontologies such as OWL and OWL-S, while adding more specialised ontology concepts to achieve precise results for automating the composition of e-services by means of abstraction. The design for life-event ontology will take OWL-S one step further to integrate atomic and composite services not only from one service provider but also from many, to allow the dynamic construction of a user-controlled workflow of readily available web services.

The model for life-event ontology requires two types of knowledge analysis in order to achieve a more comprehensive solution for the design of the ontology itself. This ontology needs to provide two essential types of knowledge about an event in a business or a persons' life.

Life-event ontology as a service knowledgebase is required to automate the acquisition of individual web service instances in a life-event workflow. It provides service-specific information such as availability, service type, service profile and required communication parameters to the runtime workflow construction process. A service knowledgebase could use multiple ontology descriptors (OWL-S ServiceModel) to obtain the semantic information required by the workflow for the invocation of atomic services. Life-event ontology embodies the following concepts:

A. Life event: a metamodel that provides a category of knowledge that is needed to answer the question 'In what possible alternative ways can a life event be constructed?' The answer to this question starts with the concept of life event that is the root element of the ontology inherited directly from the generic concept of thing. This ontology class is the definition of an abstract construct of all possible services that are nominated to collaborate with each other in order to solve a business problem. This ontology class has the inversed functional object property called hasService. This object property is of type class Life-eventService. The minimum cardinality of this property is one; this means that a life event must be composed of at least one service. One of the most important responsibilities of this class is to enforce the rules of government regulations to make sure that a legally acceptable workflow is provided that can be instantiated and executed to fulfil a customer request for a service. Listing 1 is the RDF code for the construction of this object property.

```
<owl:ObjectProperty rdf:about="le#hasService">
<rdf:type rdf:resource="&owl;InverseFunctionalProperty"/>
<rdfs:domain rdf:resource="le#Life-event"/>
<rdfs:range rdf:resource="le#Life-eventService"/>
<owl:inverseOf rdf:resource="le#describedByLife-event"/>
</owl:ObjectProperty>
```

Listing 1. Definition of object property hasService.

B. Life-eventInstance: one possible way of implementing a life event is defined by this concept, meaning that if we consider the life event as a metamodel that only defines the types and the order of possible web services in a workflow, then a Life-eventInstance

http://dx.doi.org/10.5772/intechopen.68467

would be one of the possible ways to create such workflow. This concept has an object property of type ServiceInstance called hasServiceInstance, with the cardinality of one. This property represents the individual invokeable instances of web services that make up the workflow of the life event at runtime. Listing 2 is the RDF code for the construction of this object property.

```
<owl:ObjectProperty rdf:about=”le#hasServiceInstance”>
<rdfs:domain rdf:resource=”le#Life-eventInstance”/>
<rdfs:range rdf:resource=”le#ServiceInstance”/>
<owl:inverseOf rdf:resource=”le#partOfLife-eventInstance”/>
</owl:ObjectProperty>
```

Listing 2. Definition of object property hasServiceInstance.

C. Life-eventService: provides knowledge about the acceptable web service types and possible sets of actual service instances for every service type. In other words, this concept is the abstract construction of all service types that could potentially be instantiated as a ServiceInstance at runtime. As it is strongly acknowledged by other research literature [13], the diversity of structures, regulations and procedures affecting networks of heterogeneous administrative units, represents a challenge for semantic composition. This type of knowledge is specifically related to e-government service composition, since every Life-eventService participant in any life event may enforce or be affected by one or more government regulations. These regulations are the governing rules of composite services in the e-government domain, specifically because regulations are one of the integral parts of inter-agency processes (i.e. where the life-event process flow crosses multiple agencies). Furthermore, regulatory knowledge is required for designing an inter-agency workflow that crosses the boundaries of local, state and federal agencies. It has three object properties:

1. hasPrerequisite provides the knowledge about the order of services in the workflow or possible required action or documentation prior to the invocation of the web service. Listing 3 is the RDF code for the construction of this object property.

```
<owl:ObjectProperty rdf:about=”le#hasPrerequisite”>
<rdfs:domain rdf:resource=”le#Life-eventService”/>
<rdfs:range>
<owl:Restriction>
<owl:onProperty rdf:resource=”le#hasPrerequisite”/>
<owl:someValuesFrom rdf:resource=”le#Prerequisite”/>
</owl:Restriction>
</rdfs:range>
</owl:ObjectProperty>
```

Listing 3. Definition of object property hasPrerequisite.

2. hasServiceType and (3) hasServiceSubType provide knowledge about the type of LfeEventService. Listing 4 is the RDF code for the construction of these two object properties.

```
<owl:ObjectProperty rdf:about="le#hasServiceSubType">
<rdfs:domain rdf:resource="le#Life-eventService"/>
<rdfs:range rdf:resource="le#ServiceSubType"/>
<rdfs:subPropertyOf rdf:resource="&owl;topObjectProperty"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="le#hasServiceType">
<rdfs:domain rdf:resource="le#Life-eventService"/>
<rdfs:range rdf:resource="le#ServiceType"/>
<rdfs:subPropertyOf rdf:resource="&owl;topObjectProperty"/>
</owl:ObjectProperty>
```

Listing 4. Definition of hasServiceSubType and hasServiceType.

D. ServiceInstance: it is a personalised instance of the Life-eventService and provides knowledge about the user preferences at runtime. It implements all the prerequisites of the web service that are enforced by the Life-eventService. ServiceInstance extends Life-eventService and represents one of many possible runtime instances of the Life-eventService. This ontology class has a property partOfLife-eventInstance that points to a class Life-eventInstance described in Listing 3 as the inverse functional property of hasServiceInstance. It is through this property that we can obtain knowledge about actual web services participating in a Life-eventInstance. It is our view that any problem a life event seeks to address can be fitted to two main areas of concern:

(I) For life-event users, it should describe how to ask for an OWL-S service and what happens when the workflow is being executed. By 'what happens' we mean what are the particular technical and legal requirements of invoking any of the web services in the life-event workflow.

(II) For managing the workflow, life event uses a logical description to perform four different functions:

1. To create a composite service workflow from multiple services in order to perform a specific complex task. It is important to note that 'composing OWL-S services' as a life-event is different from 'composite services' described in OWL-S specifications. The composite services described in OWL-S are provided by one service provider and specified in one WSDL grounding specification, whereas a life event comprises services from totally different providers with separate WSDL grounding specifications.

2. To manage the status and results of executing a complete or a partial life-event workflow of web services. This means that a citizen can request an invocation of a life event but does not necessarily complete the whole life event in one go. The

user also has the ability to choose to invoke any of the operations listed in a web service descriptor.

3. To coordinate the activities of different service participants (requester, provider and mediator) during the course of the web service enactment.

4. To monitor the execution of the web service and compensate for any web service failures at runtime. This is made possible by the fact that a life event is a metamodel and it can be instantiated in many possible ways depending on the availability of different actual ServiceInstances for a particular Life-eventService. There are two object properties of ServiceType and ServiceSubType that can provide essential knowledge about a particular ServiceInstance that enables a Life-eventInstance and make a decision on substituting a failed ServiceInstance with a new one that is the closest match in terms of its object properties (hasServiceType and hasServiceSubType).

The ontology illustrated in **Figure 1** displays the main concepts of life event. The abstract concept of life event is extended by Life-eventInstance. Life event is a 'metamodel' representing a generic event in a typical citizens' life. This concept points to one Life-eventService. A Life-eventService is the conceptual representation of a web service that holds information about the service type by

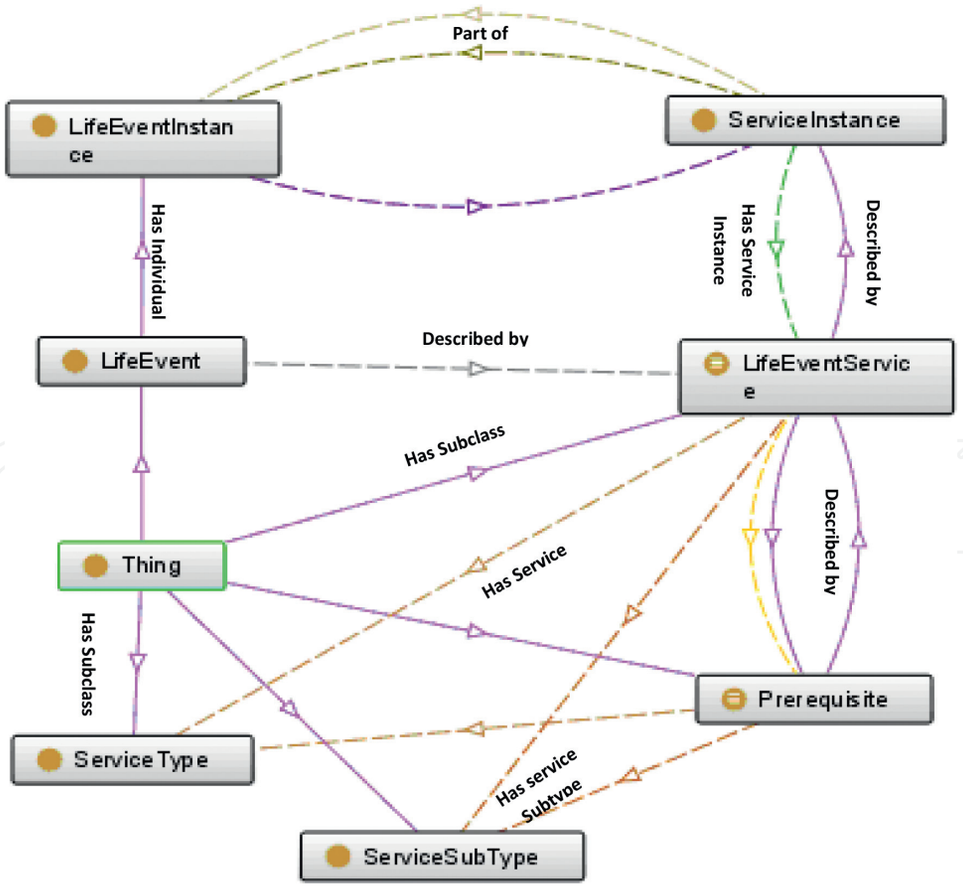

Figure 1. Life-event ontology conceptual graphs.

pointing to an instance of a class ServiceType and an instance of class ServiceSubType. Life-eventInstance is also aware of the position of its corresponding web service within the execution workflow at runtime. Life-eventService represents one element in a possible set of 'n' elements that makes up the life-event workflow.

A Life-eventService can be instantiated in many ways since it would be pointing to one or possibly more than one Life-eventInstance through ServiceType and ServiceSubType concepts.

In order to lay down a legally acceptable foundation for invocation of any government web services and to attain a legal outcome of such invocation, one needs to satisfy a set of legally binding regulatory requirements. We achieve this by setting up the rule: 'every element in this workflow must point to at least one prerequisite'. This rule is modelled as a concept called 'prerequisite'. This new concept extends the concept 'thing', therefore it could represent anything including but not limited to document, payment or, in most cases, another Life-eventService. This rule creates a linked list of services in which every Life-eventService has an object property hasPrerequisite, which is of type 'prerequisite'.

It is important to stress that the concept of prerequisite is very different from the property of 'precondition' described in OWL-S specifications; this difference is illustrated in Listing 4. OWL-S defines the property precondition to be represented as logical formulas like expressions as literals, either string literals or XML literals. The latter case is used for languages whose standard encoding is in XML such as Semantic Web Rule Language (SWRL) or RDF.

```
<Description rdf:about="#process2">
<hasPrecondition>
<expr:KIF-Expression>
<expr:expressionBody>
(!agnt:know_val_is
(!ecom:credit_card_num ?cc)
?num)
</expr:expressionBody>
</expr:KIF-Expression>
</hasPrecondition>
</Description>
```

Listing 5. Implementation of property hasPrecondition in OWL-S.

If an OWL-S process has a precondition, then the process cannot be performed successfully unless the precondition is true. The difference between the precondition properties of OWL-S and the prerequisite concept of life event falls into two major areas:

1. Consider a process that charges a credit card. The charge goes through if the precondition (card is not overdrawn) is true. If it is overdrawn, the only output is a failure notification. This means that the precondition is an expression of whether to allow the invocation of a

service to go ahead, whereas the concept prerequisite of Life-eventService extends the concept thing, therefore it could be anything, including another Life-eventService. A Life-eventService can be invoked if and only if the prerequisite is not itself and the property isFulfilled of the prerequisite service is set to true.

2. The other important difference between the two concepts is that the concept of precondition is confined within the domain of one web service, whereas a prerequisite goes beyond the domain of one web service and includes a much larger area of the universe of discourse under the domain of life event. In next sections, we will explain the significant role of the concept prerequisite where it is of the type Life-eventService. Listing 6 is a snippet of the owl tag for the class prerequisite.

```
<owl:Class rdf:about=le#Prerequisite”>
<rdfs:comment>
This class could constitute anything and is the root element of all classes
that enforce workflow regulations for the life-event ontology
</rdfs:comment>
</owl:Class>
```

Listing 6. Class prerequisite in life-event ontology.

Catering for the possibility of service substitution in runtime is made possible by the concept Life-eventService, which points to one or more ServiceInstance; this allows for the substitution of similar web services with some degree of similarity, depending on web service availability or user preferences at runtime.

5. Life-event ontology formalisation

We find it convenient to be able to speak about ontologies as objects and to have a theory of these objects. We will use a first-order language that contains the usual logical operators and symbols: for negation, for conjunction, for disjunction, $\rightarrow$ for material implication, $\leftrightarrows$for logical equivalence, $=$ for equality ($\neq$ will abbreviate its negation), for the universal and for the existential quantifier. In due course, we will introduce non-logical symbols for the relevant predicates and relations if required. We shall use x, y and z as variables ranging over existing entities, and a, b and c will be constants denoting such entities. We will use ω and ω' as variables ranging over ontologies, and α, β and γ will be constants denoting ontologies.

We do not offer a full logic, and in particular, there will be no consideration of a deductive system. For the rest, unbound variables are assumed to be within the scope of universal quantifiers.

Life-event candidate is an abstraction of a complex workflow consisting of a number of composite or simple web services. Life-event candidate can use OWL-S descriptors of web services provided by service provider including the required government regulatory information, which is required to guarantee a legally acceptable outcome whenever the service is executed.

In the following, we provide a formal description for the main concepts (classes) of our life-event ontology followed by their description logic:

Life-event = LE

Life-eventInstance = LEI

Life-eventService = LES

ServiceInstance = SI

Prerequisite = PR

ServiceType = ST

ServiceSubType = SST

We shall use the predicate 'concept' in order to denote concepts, thus 'concept(x)' is to be read, 'x is a concept', and a concept can be from any of the following types: (LE, LEI, LES, SI, PR, ST and SST). We will use x, y and z as variables ranging over concepts.

Symbolically, the first axiom is an existential one, which asserts that there is at least one entity of a certain type. Here, we indicate that there exists ontology ω and there exists entity x that is of concept (class) life event in a variable ontology ω.

$$\exists\omega, \exists x[\Omega(\omega) \wedge LE(x)] \tag{2}$$

We use the predicate Ω in order to denote token, thus 'Ω(ω)' is to be read 'ω is an ontology'. An instance of a given ontology token α is an entity whose existence is recognised by α. We will write 'inst(x, α)' which is to be read 'x is an instance of α'. Hence, there is no empty life-event ontology.

$$inst(x, \omega) \rightarrow [consept(x) \wedge \Omega(\omega)] \tag{3}$$

In addition, any existence is a constituent of ontology.

$$\Omega(\omega) \rightarrow \exists x[consept(x) \wedge inst(x, \omega)] \tag{4}$$

The predicate 'realises' denotes the instances of associated LES concepts within the life-event ontology. While each instance may be a model on its own, a combination of LESs may be aggregated to constitute a composite model. In that case, the services are considered to be the components of a model. LES (y) play role (r) that requires skills (s) needed to perform their role.

http://dx.doi.org/10.5772/intechopen.68467

$$\exists y[LES(y) \wedge (plays(y, r) \wedge has(r, s))] \tag{5}$$

Given that LES provides a set of specific operations O and x is a variable over this set to fulfil a t that is a variable over the set of tasks T, it would be required to have a subset S′ from the skill set of S.

$$[LES(y) \rightarrow (provids(y, x) \wedge (O \vee S))] \tag{6}$$

We must ensure that every LES carries enough semantic information to facilitate the runtime reconfiguration in the case of a web service failure during the LEI execution. Every LES has an object property 'hasPrerequisite' that makes one LES the prerequisite service of the value of this property in the workflow of LESs. Each of these object properties points to another LES, essentially creating a linked list of LESs in which every LES is aware of its place in the list through the data property called WorkflowPosition. In other words, a life event is the construct of a two-dimensional linked list, in which the first dimension is the list of meta-services and the second dimension is the list of service instances for each meta-service.

6. Life-event ontology evaluation

There is no restriction on the complexity of the logic that may be used to state the axioms and definitions of concepts in ontology. The distinction between terminological and formal ontologies is one of degree rather than kind. Life-event ontology tends to be smaller than terminological ontologies, but its axioms and definitions can support more complex inferences and computations. We conduct the experimental evaluation of the life-event ontology in two stages. We use the ontology editor tool Protégé to design and develop the life-event ontology schema as the preparation for evaluating the life-event ontology. We use the FaCT++ reasoner plug-in from within Protégé to perform structural validation of the schema. To evaluate the efficiency of life-event ontology, an experiment is conducted to measure the complexity of the ontology through a set of well-known formal methods and demonstrate the results in a numerical as well as graphical representation. We compare the life-event ontology to OWL-S ontology since it is the most conceptually similar to it.

6.1. Methods of measuring the ontology complexity

As ontologies grow in size and number, it is important to be able to measure their complexity quantitatively. Quantitative measurement of complexity can help ontology developers and maintainers better understand the current status of the ontology, therefore allowing them to better evaluate its design and control its development process. We are using a suite of ontology metrics [11], at both the ontology level and the class level, to measure the design complexity of life-event ontology. This ontology complexity measurement metric was evaluated in an empirical analysis on public domain ontologies to show the characteristics and usefulness of the metrics. The proposed metric suite is useful for managing the life-event ontology development projects.

6.2. Ontology level metrics

We use three different ontology level metrics to measure the complexity of the ontology:

Size of vocabulary (***SOV***) measures the amount of vocabulary defined in ontology. Given a graph representation $G = (N, P, E)$ of an ontology, where N is a set of nodes representing classes and individuals; P is a set of nodes representing properties and E is a set of edges representing property instances and other relationships between nodes in the graph G. In this measurement, SOV is defined as the cardinality of the named entities N_n and P_n in G: $SOV = |N_n| + |P_n|$, where N_n represents named classes and individuals and P_n represents user-defined properties.

Edge node ratio (***ENR***) measures the connectivity density. In this measurement, ENR tends to increases as more edges are added between nodes. The greater the ENR, the greater the complexity of an ontology. ENR is calculated as follows:

$$ENR = \frac{|E|}{|N|}, \tag{7}$$

as the division of the number of edges ($|E|$) by the number of nodes ($|N|$).

Tree impurity(***TIP***) measures how far ontology's inheritance hierarchy deviates from being a tree and it is defined as being:

$$TIP = |E'| - |N'| + 1, \tag{8}$$

where $|E'|$ is the number of *subclass* edges and $|N'|$ is the number of nodes in an ontology's inheritance hierarchy.

6.3. Class level metrics

This metrics are mostly concerned with the class level specific statistics, the most popular technique in this method is known as ***number of children*** (***NOC***), as such to calculate NOC for a given class C, NOC measures the number of its immediate children in the ontology inheritance hierarchy given, as follows:

$$NOC_c = \#\left\{D | D \in N' \wedge (D, rdfs : subClassOf, C) \in E'\right\}, \tag{9}$$

where $C \in N'$ and symbol # denote the cardinality. And, the E' denotes the set of entities.

6.4. Experiment preparation

This section will describe the preparation for a comparative evaluation of life-event ontology against OWL-S using the methods described in Sections 6.2.2 and 6.2.3.

Step 1: Building the ontology. It aims to prepare for the evaluation of the life-event ontology. The choice of ontology editor was made mainly due to the fact that Protégé is open source software and was more suited to our purpose [14]. This tool is developed and maintained by

http://dx.doi.org/10.5772/intechopen.68467

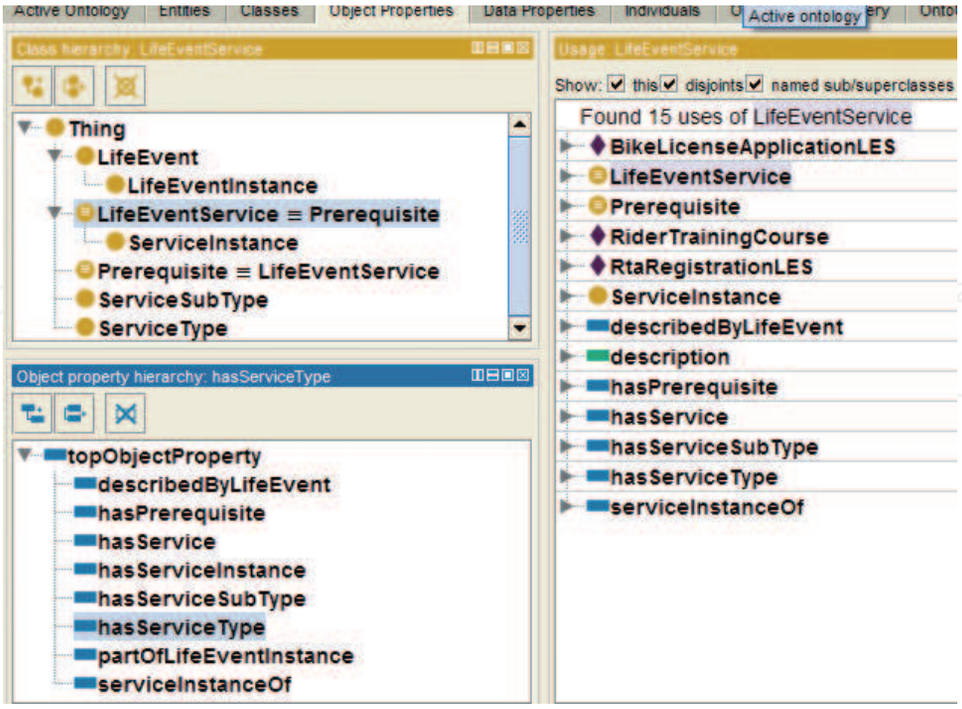

Figure 2. Life-event ontology built by Protégé.

Stanford University. We used version 4.1, which was more advanced, intuitive and easier to use than other available ontology editors. **Figure 2** shows a screenshot of our developed ontology, which illustrates the concepts, their relationships with object properties and data properties in the Protégé ontology editor.

Step 2: Comparable ontology. This step is to select an ontology that is conceptually similar to life-event ontology. OWL-S is chosen because it is not only conceptually very similar to the life-event ontology but also functionally designed to perform a similar work. This ontology is also used by the LOOSI platform to provide knowledgebase support for the web service enactment functionality of the system. The diagram shown in **Figure 3** is the graph representation of OWL-S conceptual schema version 1.1.

In this comparative evaluation of life-event ontology with OWL-S, we use numerical value results from applying the metrics in Sections 6.2.2 and 8.2.3 on both ontologies to illustrate the measurement of efficiency and complexity of the life-event ontology in compare to the OWL-S.

The experiment starts with creating and adding five named individuals that are the representatives of five individual web services that are published by the Australian Government agencies and other businesses. One more named individual is created only in life-event ontology as the first instance of the schema to point to the Life-eventServices. The list of these named individuals is described in **Figure 4**.

Considering the populated OWL-S ontology and the life-event ontology, we use the actual measurements with the methods described in Sections 6.2.2 and 8.2.3 and calculate the results as follows:

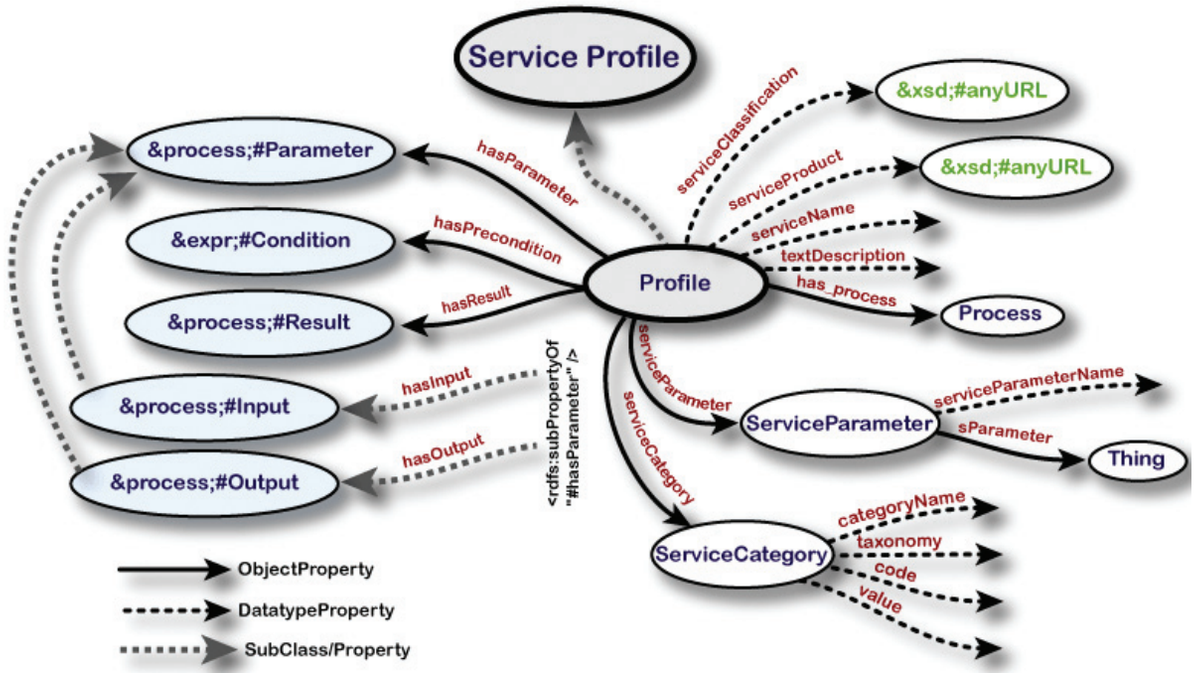

Figure 3. OWL-S ontology schema [10].

Based on the SOV method, we measure the SOV value of the life event to be $7 + 8 = 15$, and for OWL-S to be $12 + 9 = 21$, this means that if we consider the growth ratio for the life event as being the base $15/15 = 1$ then this for the OWL-S would be $21/15 = 1.4$. **Table 1** details the numerical representation of the growth of ontology data in both ontologies.

We register the statistics in **Table 1** by assuming the initial SOV as to be the sum of the nodes, plus object properties in the ontology schema. Then we increased this number five times as per the number of named individuals representing the web services created for this experiment, each time by the amount of SOV ratio, representing the linear growth in the volume of ontology data.

Figure 5 shows the comparative graph representation that illustrates the trend of growth in the life-event ontology data and the OWL-S ontology data. It is shown that the rate of growth in the volume of data in the life event is dramatically less than the OWL-S, after a fivefold increases in the number of named individuals.

a. Based on the ENR method, we measure the ENR value of both ontologies in question to be as follows:

$$\text{Life} \quad -\text{event} = \frac{|15|}{|7|} = 2.5, \text{◎OWL} \quad -\text{S} = \frac{|21|}{|8|} = 2.63. \tag{10}$$

Table 2 shows the growth of ontology data in both ontologies in terms of ENR in numerical terms. The statistics shown in **Table 2** is obtained by initial ENR 1 is increased five times as per the number of web service named individuals, created for this experiment, each time by the amount of ENR ratio.

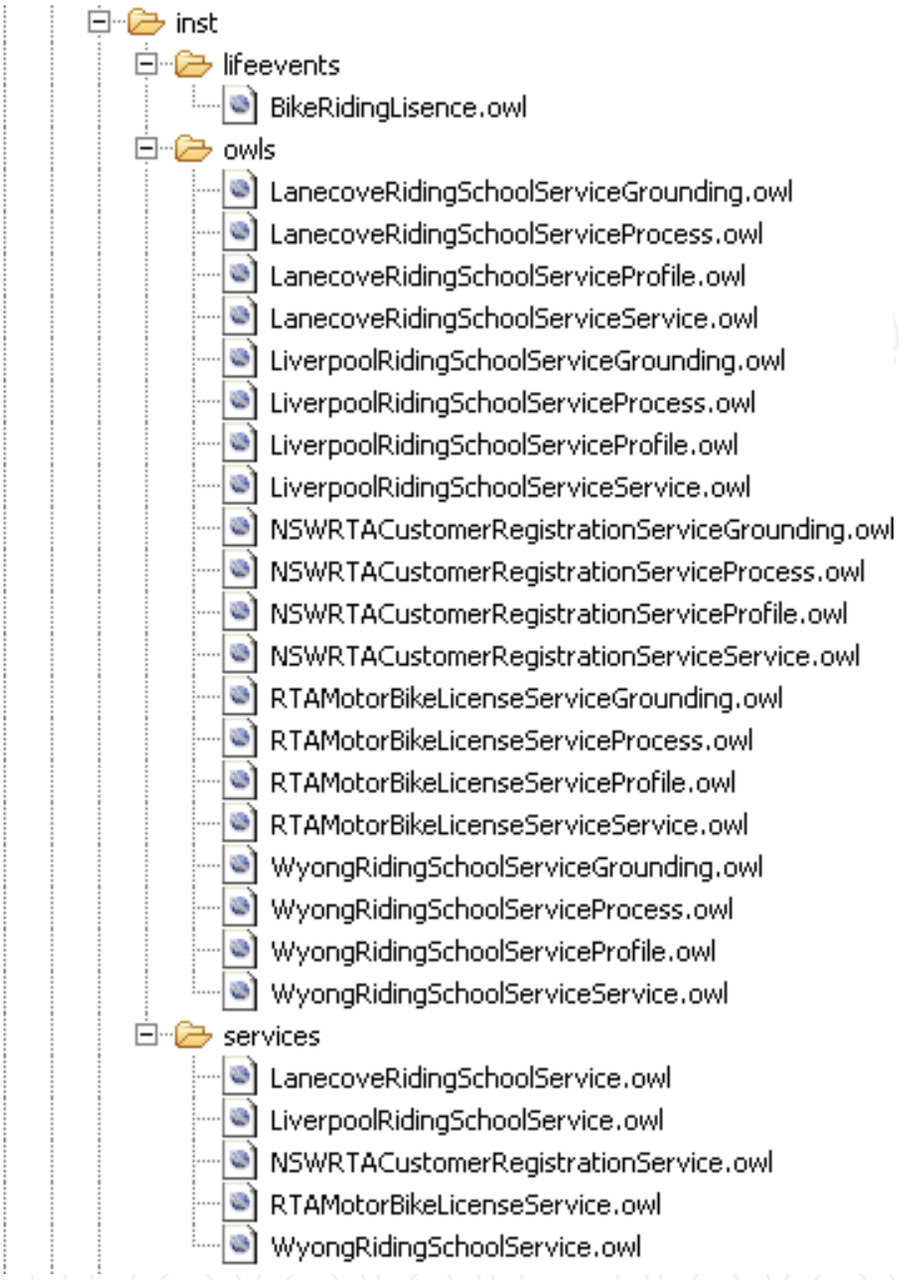

Figure 4. Life event and OWL-S named individuals.

Life event	OWL-S
15	21
30	36
60	86.4
120	207.36
240	497.7

Table 1. Numerical representation of ontology growth as per SOV ratio.

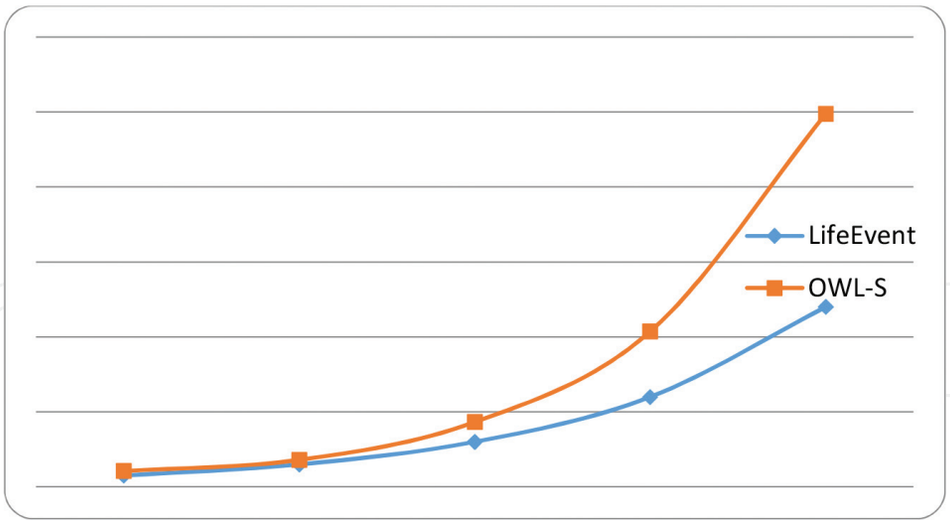

Figure 5. Physical representation of growths in ontology data as per SOV ratio.

Life event	OWL-S
2.5	2.63
6.25	6.9
15.63	18.2
39.1	47.8
97.75	125.8

Table 2. Numerical representation of physical growth for ontology as per ENR ratio.

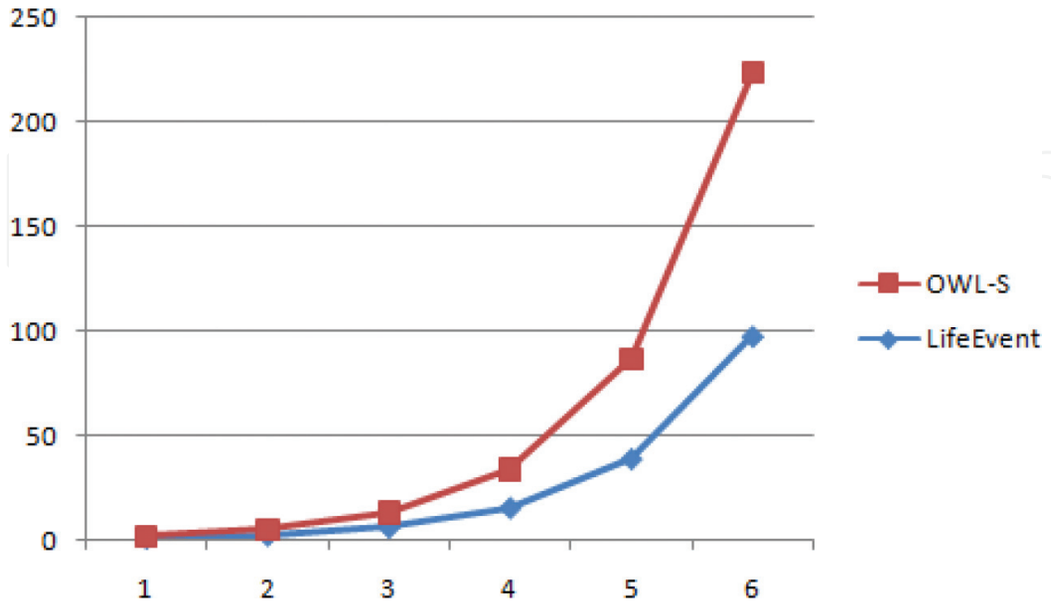

Figure 6. Physical representation of growths in ontology data per ENR ratio.

Figure 6 shows the comparative graph illustrating the trend of growth in ENR for the life-event ontology and OWL-S ontology in the event of growth in ontology data. It is shown that this increase in life-event ontology is less than OWL-S after a few fold increases in the number of named individuals.

b. Based on the TIP method, we measure the value of 'how far life event deviates from being a tree?' to be $(15 - 7 + 1 = 9)$, and for OWL-S to be $(21 - 8 + 1 = 14)$. As it is shown that this value is greater for OWL-S than for life-event ontology.

c. Using the NOC method, we calculated the number of immediate children (*rdf: subClassOf*) for the class *Parameter* that is the most frequently used in web service invocation to be 3. The value of NOC calculated for Life-eventService, which is the most used class in life-event ontology, is 2. **Table 3** shows the complexity growth of ontology data for class parameter in OWL-S in comparison with the class Life-eventService in life-event ontology in terms of the NOC ratio. The statistics shown in **Table 3** is obtained by initial NOC 1 increased five times as per the number of web service named individuals, created for this experiment, each time by the amount of NOC ratio.

Life event	OWL-S
2	3
4	9
8	27
16	81
32	243

Table 3. Numerical representation of physical growth for ontology classes as per NOC ratio.

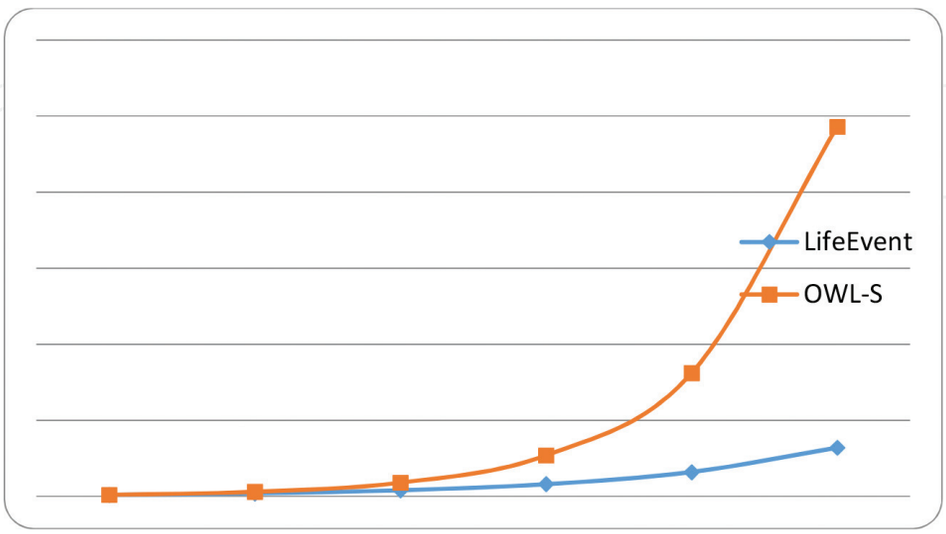

Figure 7. Physical representation of growths in complexity of ontology data as per NOC ratio.

Figure 7 shows the comparative graph illustrating the trend of growth in complexity as per the NOC ratio for class parameter in OWL-S in comparison with the class Life-eventService in life-event ontology in the event of growth in ontology data. It is shown that this increase in life-event ontology is less than OWL-S after a few fold increases in the number of named individuals.

7. Summary

In this chapter, we outlined our design strategies to extend the OWL-S in order to propose life-event ontology as an abstract design and execution unit for composing e-services. We introduced an ontology that accommodates the concept of life event within the process of e-services composition. The idea was to introduce an innovative approach towards the whole process of e-service composition and delivery.

We put forward a formal design for an ontology knowledgebase to manage the workflow of composite web service workflows in a linear approach. Nevertheless, this research also recognises that more research is required to specify and formalise the design of more complex types of web service composition such as parallel service processes in complex workflows.

Author details

Farzad Sanati

Address all correspondence to: f_sanati@yahoo.com

The American University of Iraq, Sulaimani, Iraq

References

[1] Sanati, F. and J. Lu, An ontology for e-government service integration. International Journal of Computer Systems Science And Engineering. 2012;**27**(2):89-101

[2] Smith MK, Welty C, McGuinness DL. Owl Web Ontology Guide. 2003 [cited 2011]

[3] Berners-Lee T, Hendler J, Lassila O. The semantic web. Scientific American. 2001;**2845**:34-43

[4] Maedche A. Ontology Learning for the Semantic Web. Kluwer International Series in Engineering and Computer Science; SECS665. Boston: Kluwer Academic; 2002

[5] Berners-Lee T, Fielding R, Masinter L. Uniform Resource Identifiers (URI): Generic Syntax. RFC Editor 1998

[6] Valencia-Garcia R, et al. OWLPath: An owl ontology-guided query editor. IEEE Transactions on Systems, Man, and Cybernetics—Part A: Systems and Humans. 2011;**41**(1):121-136

[7] McGuinness DL. OWL Web Ontology Language Overview [Internet]. 2004. Available from: http://www.w3.org/TR/2004/REC-owl-features-20040210/

[8] Tsarkov D, Horrocks I. FaCT++ description logic reasoner: System description. In: Proceeding of the International Joint Conference on Automated Reasoning (IJCAR 2006). 2006. Springer

[9] Koponen T, Virtanen T. A service discovery:A service broker approach. In: 37th Annual Hawaii Internation Conference on System Science. 2004: Hawaii, USA. pp. 284-290

[10] W3C. OWL-S: Semantic Markup for Web Services. 2004 [16 November 2010]

[11] Zhang H, Li Y-F, Tan HBK. Measuring design complexity of semantic web ontologies. Journal of Systems and Software. 2009;**83**(5):803-814

[12] Ferrario R, Guarino N. Towards an ontological foundation for services science. In: Future Internet Symposium. 2008. Springer

[13] Lytras MD, García R. Semantic web applications: A framework for industry and business exploitation—What is needed for the adoption of the semantic web from the market and industry. International Journal of Knowledge and Learning. 2008 Jan 1;**4**(1):93-108

[14] Stanford-University. Protégé. 2009. Available from: http://protege.stanford.edu/. [Accessed: 2010]

IntechOpen

IntechOpen

IntechOpen