Simulation Oriented OWL-S Extension and Cloud Interaction

Zhi-hui JIANG [1,*], Jian-feng LU [1], Hui WANG [1] and Yao-ping YU[1]
[1]No. 4800, Cao an Road, Jia ding District, Shanghai, China

Keywords: OWL-S, Evaluate, Web service, Service composition, Model interaction

Abstract. In the field of cloud manufacturing, in order to evaluate cloud service composition better, simulation could be introduced into the process of cloud service composition. This job requires adding simulation information to the service description document. Besides, for achieving the simulation of cloud services, the interaction, transmission problems of simulation model must be solved. This paper proposed a method to evaluate the service composition through simulation, and studied the methods for injecting simulation model information into OWL-S, OWL file parsing method based on Jena API, and provided a solution for the interaction problem of simulation model. Using a factory simulation platform"Plant Simulation", this paper provided an example which used the model data given by OWL file to generate a simulation model dynamically.

Introduction

Cloud manufacturing is a new model of network manufacturing, which organizes online manufacturing resources in accordance with the users' needs, and provides users with all kinds of on-demand manufacturing services using network and cloud manufacturing service platform [1]. In all kinds of activities of the product manufacturing life cycle, cloud manufacturing pays more attention and emphasis on some needed services [2], such as simulation as a service (SimaaS). Simulation has been an indispensable technology and tool in manufacturing process [3]. Cloud manufacturing has put forward new requirements to the simulation, that is, to achieve a new model of cloud simulation [4].

According to the granularity of user's requirements, requirements can be divided into requirement for single resource service and requirement multi resources service. For multi resources services, system must select cloud services from several candidate service sets to assemble into a combination of cloud services and select a best combination from all possible combinations to complete the task [5]. It is necessary to select higher quality services to form a larger granularity service. The search and selection of Web services based on service quality for improving the quality of service composition has become one of the hot research questions [6, 7].

It is difficult to describe a complex system by analytic method. Also the analytic model is not easy to construct for a complex system. If the model is simplified, the system may not be described accurately. This paper studied simulation method to evaluate composite service. Compared with the analytic method, simulation is more accurate and reliable in evaluating the processing capacity of manufacturing services.

As shown in Figure 1, in the phase of service publication, the simulation mapping information is injected into the service description file. This paper use OWL-S to describe Web service. OWL-S supports automatic discovery, invocation and composition of Web services through agent based on logic semantics. After the service

composition, generating a simulation model automatically, according to the composite service. And then, the composite service can be evaluated by the simulation result.

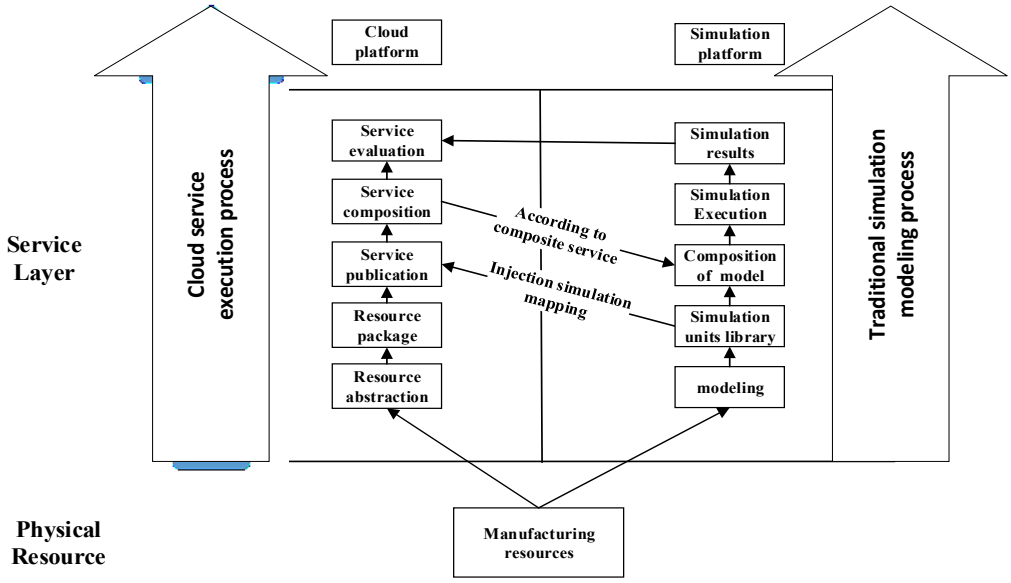

Figure 1. General framework

Web Services Description Language

The W3C Web Ontology Language (OWL) is a Semantic Web language designed to represent rich and complex knowledge about things, groups of things, and relations between things. OWL not only can be compatible with the RDF(s), but also can describe ontology formally. Compared to other languages, OWL has more abundant basic modeling elements, such as inverse, symmetric and transitive, through which OWL has more accurate semantic reasoning ability.

OWL-S is Web service ontology language based on OWL, which defines a set of core components of language to describe Web service logically, so that OWL-S document can be understood by machine and can be used to discover or combine Web services. OWL-S has three main parts: the service *profile* for advertising and discovering services; the *process model*, which gives a detailed description of a service's operation; and the *grounding*, which provides details on how to interoperate with a service, via messages.

Some Web services are described by WSDL. These WSDL files can be transformed to OWL-S by OWL-S editor easily. This paper studies how to add information needed for simulation to OWL-S.

OWL-S extension

Figure 2 shows a simple Plant Simulation model. The "Source" module is used to generate the parts to be processed, the "SingleProc" module represents processing station, the "Drain" station receives parts which have been processed. Every station has its own method property and processing time property.

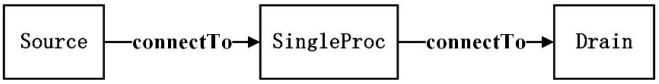

Figure 2. Plant Simulation model

The simulation model mentioned above can be constructed as ontology by protégé which is a kind of ontology editing tool and invented by Stanford University. The simulation model ontology constructed by protégé is shown in Figure 3 below.

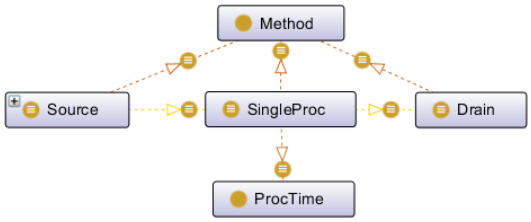

Figure 3. Simulation model ontology

Service grounding is one of main parts in OWL-S which can be thought of as a mapping from an abstract to a concrete specification of those service description elements that are required for interacting with the service. This study extends OWL-S and associates with the simulation model ontology in service grounding. The structure diagram after extension is shown in Figure 4:

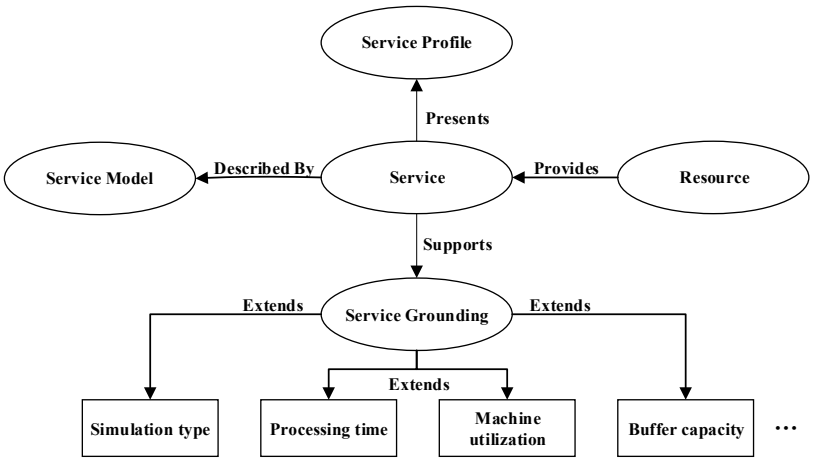

Figure 4. OWL-S extended structure diagram

OWL-S Document Parsing

The composite service after simulation ontology expansion is described by OWL-S as well. This OWL-S document contain simulation information in the service grounding. The simulation information can be got by parsing the OWL-S document.

This paper studies using Jena to parse the OWL-S document. Each of the arrows in the ontology model shown in Figure 3 is a statement in the Jena. Statement is composed of three parts, namely, subject, predicate and object. The subject is shown in the arrow starting position, on behalf of resources; the predicate is just the arrow representing resources' attributes; the object is shown in the position that the arrow points to, on behalf of the property, which could be texts or resources.

Specific OWL document parsing steps are as follows:

1) Creating a memory model using the OWL language in memory;
2) The OWL document is read into the ontology model object;
3) Extracting Statement and its subject, predicate and object in the ontology model;
4) Storing model information in database.

Example

Simulation Ontology Extension for OWL-S

As is shown in Figure 5, assuming that an enterprise published a production line service on the cloud platform. At the beginning, this Web service may be described by WSDL. The OWL-S document could be got by the way of transforming WSDL file. And this Web service could be regarded as a simulation model in Plant Simulation as well. The input device, could be named *Source* in Plant Simulation, which is used to generate parts to be processed. The module named *upright* is a kind of upright station which is a *SingleProc* in Plant Simulation, the same as *clean* module which is a cleaning station. The output device, regarded as *Drain* in Plant Simulation, represents a kind of station which receives processed units. IO devices and all processing stations have their own property such as running time and processing method. Through this simulation model, users can know the processing time of the entire manufacturing process and simulate running state of the production line. And then, users will know whether the Web service is suitable for them or not.

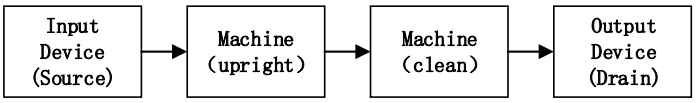

Figure 5. Production line service

By Jena API, protégé or other means, the simulation model can be transformed to an OWL file. Because contents of every method property contain spaces, enter and other format information, they cannot be stored directly in the ontology file as a resource. In this paper, these methods will be stored locally in the text documents. And path to text document will be encapsulated as a resource, rather than the method itself. Paths of methods' text files of four stations in Figure 5 are "E:/source.txt", "E:/upright.txt", "E:/clean.txt" and "E:/drain.txt". The running time property of upright and clean stations, called ProcTime, are 30 and 60. In this paper, the ontology of simulation model had been built by protégé and added to service grounding of OWL-S by OWL-S editor. Some segments are as follows:

1) Adding node of simulation information to service grounding in OWL-S:

```
<owl:Class
rdf:about="http://www.daml.org/services/owl-s/1.2/Grounding.owl#SimInfo">
    <rdfs:subClassOf
rdf:resource="http://www.daml.org/services/owl-s/1.2/Grounding.owl#Grounding"/>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:someValuesFrom rdf:resource="#SInfo"/>
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="hasSInfo"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>
```

2) Adding paths of methods and connection information between stations in OWL-S:

```
<Source rdf:ID="Source1">
    <connectTo rdf:resource="#upright"/>
```

```
    <hasMethod rdf:resource="#E:/source.txt"/>
  </Source>
```

3) Adding time property of upright and clean stations:

```
<owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
>60</owl:cardinality>
<owl:onProperty rdf:resource="#hasProcTime"/>
```

Example for OWL Document Parsing

This step help us to get simulation information from the OWL file which named ProductionLine.owl.

1) Creating a memory model using the OWL language in memory:

```
OntModel ontModel =
ModelFactory.createOntologyModel(OntModelSpec.OWL_MEM);          (1)
```

ModelFactory class, as a model factory, is used to create ontology model objects

2) The OWL document is read into the ontology model object:

```
File file = new File("E:/ProductionLine.owl");                   (2)

FileInputStream fis = new FileInputStream(file);                  (3)

ontModel.read(fis, null);                                         (4)
```

3) Extracting Statement and its subject, predicate and object in the ontology model:

By using the listStatements method in OntModel class, obtaining the statement iterator (iter). And using the getSubject, getObject method to get the subject and object.

```
Statement stmt = iter.nextStatement();                            (5)

Resource subject = stmt.getSubject();                             (6)

RDFNode object = stmt.getObject();                                (7)
```

4) Storing model information in database:

Storing the resource information in tables (Table 1 and Table 2) in the database, Plant Simulation can get information from these tables for construction of model.

Table 1. Model attributes

id	station	time	method
1	Source	-	E:/source.txt
2	upright	30.0	E:/upright.txt
3	clean	60.0	E:/clean.txt
4	Drain	-	E:/drain.txt

Table 2. Model connection information

id	FromStation	property	ToStation
1	Source	connectTo	upright
2	upright	connectTo	clean
3	clean	connectTo	Drain

Example for Construction of Model in Plant Simulation

In order to meet the needs of the operating database, Plant Simulation has developed multiple database interfaces, such as Oracle, SQLite and ODBC interface. This paper uses the SQLite database.

For operating database in Plant Simulation, the first thing is to check the SQLite module in the "Manage Class Library". After connecting to the database through the SQLite module, using ModelLibrary's createObject function to generate all kinds of stations, also adding corresponding simulation time and method in these stations. At last, the model in Figure 6 will be got. This model can be used in simulation of manufacturing process and can simulate running state of production line. The output of this model is the time spent in the whole process. Simulation results have been recorded in Figure 7, which can be used for service evaluation. Users will know whether the manufacturing service can complete their task on time and enterprise will know resource utilization rate, bottleneck analysis and other information.

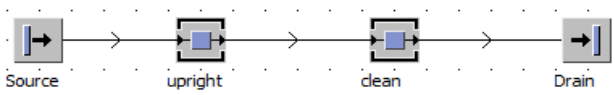

Figure 6. Plant Simulation model

Simulation time: 1:30.0000

Cumulated Statistics of the Parts which the Drain Deleted

Object	Name	Mean Life Time	Throughput	TPH	Production	Transport	Storage	Value added	Portion
Drain	Entity	1:30.0000	1	40	100.00%	0.00%	0.00%	100.00%	

Figure 7. Simulation results

Conclusions

(1) This paper proposes a method to evaluate the service composition by simulation and studies to inject simulation model information into OWL-S.
(2) The parsing method for OWL document had been studied to solve the cloud interaction of simulation model.

Acknowledgement

This research was financially supported by the National Science Foundation (No. 61273046).

References

[1] Li B H, Zhang L, Wang S L, et al. Computer integrated manufacturing systems, 2010, 16(1): 1-7(In Chinese).

[2] Li B H, Zhang L, Ren L, et al. Computer integrated manufacturing systems, 2011, 17(3): 449-457(In Chinese).

[3] Xiong G L, Wang X. Journal of system simulation, 1999, 11(3): 145-151(In Chinese).

[4] Yang C, Li B H, Chai X D, et al. Computer integrated manufacturing systems, 2012, 18(7): 1444-1452(In Chinese).

[5] Tao F, Zhao D M. IEEE Transactions on industrial informatics, 2008, 4(4): 315-327.

[6] Feng D G, Zhang M, Zhang Y, et al. Journal of software, 2011, 22(1): 71-83.

[7] ZAKI M, ATHMAN B. IEEE Internet Computing, 2009, 13(1): 40-47.