Research and Engineering Design in CPN of OWL-S Service

Ziyun DENG\(^1\)\(^2\)*, Jing ZHANG\(^1\), Lijun CAI\(^2\)\(^3\), Lei CHEN\(^1\)\(^2\) and Xiaolin LIANG\(^2\)\(^3\)

\(^1\)Electrical and Information Engineering, Hunan University, Changsha, Hunan, 410082, China  
\(^2\)National Supercomputing Centers in Changsha, Changsha, Hunan, 410082, China  
\(^3\)Information Science and Engineering, Hunan University, Changsha, Hunan, 410082, China  
* dengziyun@126.com

Abstract

A description language CPSWS (Colored Petri net markup language of Semantic for Web Services) is suggested, which improved the formal definition of CPN (Colored Petri Net) of OWL-S (Ontology Web Language for Services) service. The language combines some abilities including semantic representation of OWL-S, automatic service combination of OWL-S, formal definition of CPN, and service description of CPN. The overall idea of engineering design is given, SCAToOWL algorithm which used to converted SCA component and composite into CPSWS is proposed in the paper. Research work about the paper is verified by the case which is the process of the job computing in the cloud platform for industrial design and simulation. Finally the direction for improving is put forward, including the realization of describing function, the integration technology to open source Tuscany and Spring.

Keywords: OWL-S, CPN, CPSWS, SCAToOWL algorithm, Process of job computing

1. Introduction

The basic purpose of Web Service combination is to organize Web Services by certain logical way so as build more feature-rich Web software with SOA (Service-Oriented Architecture) \[1\]. Web Service combination have become core technology of SOC (Service-Oriented Computing), which is one of the main directions of integration technology for development loosely coupled software under distributed environment\[2\]-\[4\], many scholars and engineers are now working on related scientific theory and engineering technology\[2\]-\[17\].

With the explosive growth of Web Services in Internet and cloud computing platform, it’s difficult for researchers to identify, select, combine, and verify Web Services and their combination. Using and optimizing some of the existing technology is a way to study a combination of Web Services technology, such that Web Services can be combined automatically \[2\]-\[6\], \[13\]. How to describe Web Services to make them easier to be combined, and gradually move toward automated combination, it is the key problem for researcher to study Web Services combination technology \[3\]-\[4\], \[7\].

Currently, some researchers have made some achievements on service combination technology and Web Services description, they are relatively more concentrated focus on how to optimize the various properties of description service to enhance the service combination capabilities in generic or specific environments, such as increasing the QoS (Quality of Service) description, semantic description, and using various combination technology to increase the content of some combination of capacity and automatic. The main contribution of our works is summarized as follows.
We formally define Web Services and combination Web Services by using CPN with OWL-S so as to have the automatically combined capacity. Considering PNML design ideas, we design CPSWS language based on XML (eXtensible Markup Language) for describing Web Services and combination Web Services with CPN of OWL-S.

We still use WSDL as description file to implement engineering of Web Service, the open source software Tuscany as a container for Web Services, Spring as a container for combination beans and combination Web Services. Design SCAToOWL algorithm which used to automatically convert description file of Tuscany container components and composites to CPSWS description file. Finally, we verified our work by designing a job computing case in the cloud computing platform for industrial design and simulation.

2. Related Work

In this section, we will introduce related work on three aspects including combination technology of Web Services, comparison about combination technology of Web Services and the description method of Web Services.

2.1 Combination Technology of Web Services

Existing research of Web Service combinations are mainly focused on two aspects [1, 12], the aspects are combination method of Web Services based on artificial intelligence theory and combination method of Web Services based on business process-driven.

Combination method of Web Services based on artificial intelligence theory can be abstracted into automatically solving solution of programming problem, the core of the problem is how to find a service path from one service set begin with the initial state to the target state [4, 8, 9, 14]. This method mainly has hierarchical task network planning, situation calculus, rule-based planning [14], etc.

Combination method of Web Services based on business process-driven can be divided into two kinds [3, 13], one is based on workflow, and other is based on formal method.

Combination method of Web Services based on workflow include three main methods, BPEL4WS/BPEL (Business Process Execution Language for Web Services), BPML (Business Process Modeling Language), WS-CDL (Web Services Choreography Definition Language) [13]. Combination systems of Web Services based on workflow include METROR-S, eFlow, SCM, SELF2SERV, Activiti, JBPM, etc. [16, 17]

Web Service combination based on formal methods concentrated on Petri Net, Process Algebra [1], etc., Petri Net can describe concurrency conflicts and synchronization process; can graphical description and having formal definition [16]. Many researchers dedicated to improving Petri Net technology, and applied to Web Service combination [1, 3, 4, 10, 16]. Process Algebra includes CCS (Calculus of Communication System), CSP (Communicating Sequential Processes), Pi algorithm, and there are some theoretical researches and some engineering application cases [2, 8, 10, 13], as for the analysis of whether the service can be replaced by another service.

2.2 Comparison ABOUT Combination Technologies of Web Services

Combination technologies of Web Services based on artificial intelligence theory method is not intuitive, rigorous and practical than business process-driven method [2, 5, 12-14], the difference of Web Service combination will show on Table 1.

As shown in Table 1, BPEL doesn’t support dynamic combination and automatic combination [1, 12, 16], which is lack ability to describe non-functional properties; OWL-S has the weak performance of exception handling and compensate ability; Petri Net and Process Algebra are more comprehensive in automatically combined technology, but they are difficult applied to project implementation and applications. Therefore, considering
the various technologies and the weaknesses, we will focus on combination used OWL-S and Petri Net to more fully describe the Web Service and support automatic combination.

### Table 1. Comparison about Combination Technologies of Web Services

<table>
<thead>
<tr>
<th>Items</th>
<th>Static combination</th>
<th>Dynamic combination</th>
<th>Automatic combination</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BPEL</td>
<td>OWL-S</td>
<td>Automata</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Petri Net</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Process Algebra</td>
</tr>
<tr>
<td>Connectivity expression</td>
<td>√</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>Non-functional attributes description</td>
<td></td>
<td></td>
<td>√</td>
</tr>
<tr>
<td>Correctness verification</td>
<td></td>
<td></td>
<td>√</td>
</tr>
<tr>
<td>Automatic combination</td>
<td></td>
<td></td>
<td>√</td>
</tr>
<tr>
<td>Exception handling mechanism</td>
<td>√</td>
<td></td>
<td>√</td>
</tr>
<tr>
<td>Support Tools</td>
<td></td>
<td></td>
<td>√</td>
</tr>
</tbody>
</table>

The researches about enhance the ability to describe Web Services by using variety of technologies and supports automatic or semi-automatic combination have made considerable progress [4-8, 12, 14-16], such as the METROR-S developed by the Georgia University LSDIS (Large Scale Distributed Information Systems) laboratory [18] to achieve Web Service combination experience BPEL4WS control process modeling, service description template requirements, process optimization, service matching of service engine, service selection, Web process conversion process. They use the OWL-S for Web Services semantic description among these processes.

### 2.3 Research in Web Service Description

In paper [19], the authors proposed a service analytical model, they point out the key properties of atomic services are these properties, service interface, I/O (Input / Output) message, parameter type, execution time, parent node, ancestor node, combination rule, etc., so as to promote the realization of the service combination by the description of these properties. In paper [20], the authors proposed a description model of Web services based on QoS, which mainly focus on QoS indicators including utilization performance, reliability, cost, etc. In paper [21], the authors defined a description model of Web Services which can describe three aspects including content, function and property.

Many scholars have proposed the Integration of RDF (Resource Description Framework) [22] and ontology to increase the semantic representation on service description. In paper [23], the authors put forward a service description model which contains functions, performances, and semantic information based on combination of UDDI (Universal Description Discovery and Integration), WSDL (Web Services Description Language), and OWL-S.

Most Web Services select WSDL as describe language, which is a standard language for engineering implementation of XML-based Web Service [6]. UDDI is usually used for standard service center which provides language-based for Web publishing service, discovery, and integration [17]. SCA (Service Component Architecture) has become a member of the OASIS (Organization for the Advancement of Structured Information Standards) [10], which is a described standard for describing components and composites. SCA can bind for Web Service, JMS (Java Messaging Service) and other types of services.
through SCA component or composite engine. China National Institute of Standardization and other units jointly drafted the "common services and service interface description specification (draft)" defines the CSDM model (Common Service Description Model), having the description of common services from four main aspects like service summaries, interfaces, qualities and strategies [24].

In paper [5], the authors proposed service behavior model based on OWL-S, and the model has a formalized description about behavior for Web Services. In paper [11], the authors designed a model called SODM (Service Opening and Description Model) to describe Web Services and description the service through eight basic attributes such as the service profile, function, interface, size, status, quality, transaction, security.

In order to better support automatically generates the Web Service combination used Petri Net, In paper [25], the authors proposed a method which establishes PNML (Petri Net Markup Language) and OWL (Ontology Web language) of Web Services at the time of publishing Web Services, adds the appropriate semantic markup for Petri Net library elements, and gets similar service description to Web Services of OWL-S. There is no research about described SCA files into PNML files.

3. Semantic Web Services and Petri Net

Some basic terminologies and knowledge points about work thinking of this paper are respectively elaborated below.

3.1 Semantic Web Services

The emphasis of semantic Web research is how to convert information into a form which can be understood and deal with by computers, namely with semantic [25]. Semantic Web Services is the combination of semantic Web and Web Services, and it is Web Services based on the semantic intelligent. The main research content of semantic Web Services is Web Service automation, including automatic search, automatic execution, automatic combination, automatic call, etc. [25].

OWL-S is the Web Service ontology with OWL language describing, and it is a language in common use of semantic description languages of Web Services [26]. The ontology structure of OWL-S includes three parts, which are service profile, service model and service basis [26].

Service Profile of OWL-S is used to describe what service does. The framework is as shown in the Figure 1. The main describing information is IOPE (Input/Output/Precondition/Effect) [25, 26]; the others can also have service non-functional description, such as service classification, QoS, etc. Service profile can also include some custom description. Service model mainly describes how service does, namely the composition situation of service internal, the non-functional description of the services, for example, the category of services and the quality information of services. Service basis describes how to access the services. However, OWL-S self about description of Web Service is not perfect, since it needs to make appropriate improvements according to specific application field or user requirement.

![Figure 1. Service Profile of OWL-S](image-url)
In OWL-S, atomic service is indivisible service; Compound service consists of several atomic services and compound services, and it possesses the procedural model of service combination, the execution sequence of the services, the execution processes of the services, etc.

3.2 Petri Net

When using Petri Net to model system especially concurrent systems, Transition used to express event, action, conversion, etc., Place represents such as condition, resource, status, Token shows the number of resources and connecting arc qualitatively denotes the link between Place and Transition (consumption or production) [27]. As shown in Figure 2, some related definitions are derivative.

![Petri Net](image)

**Figure 2. Petri Net**

**Definition 1: Direct Net [28]**

If triple \( N = (P, T, F) \) meets the following conditions, it is a Direct Net.

1) \( P \cup T \neq \emptyset, \ P \cap T = \emptyset \)
2) \( F \subseteq (P \times T) \cup (T \times P) \)
3) \( \text{dom}(F) \cup \text{cod}(F) = P \cup T \)

Among them,

\[ \text{dom}(F) = \{ x \in P \cup T \mid \exists y \in P \cup T: (x, y) \in F \} \]
\[ \text{cod}(F) = \{ x \in P \cup T \mid \exists y \in P \cup T: (y, x) \in F \} \]

Conditions 1 represents Place and Transition are the basic elements of the net \( N \). They belong to two disjoint collections, but one element is in the net at least.

Condition 2 represents \( F \) is a set of network connecting arcs, only including the connecting arcs from a Place to a Transition and from a Transition to a Place, not including the connecting arcs from a Place to a Place and from a Transition to a Transition.

Condition 3 represents a Direct Net needs not have the isolated Place or Transition. All Places and All Transitions are in the network.

The definition of Direct Net is the basis of Petri Net further definition.

**Definition 2: Pre Set, Post Set, Extension [28]**

In the Direct Net \( N \) denoted by the definition 1, \( x \in P \cup T \).

\[ x = \{ y \mid y \in P \cup T \land (y, x) \in F \} \]
\[ x^* = \{ y \mid y \in P \cup T \land (x, y) \in F \} \]

\( x \) is called as \( x \) Pre Set or Input Set, \( x^* \) is called as \( x \) Post Set or Output Set, \( x \cup x^* \) is called as \( x \) Extention.
Definition 3: Petri Net [27-28]

If five tuples \( PN = (P, T, F, M, M_0) \) meets the following five conditions, it is called as a Petri Net.

1) \( (P, T, F) \) is a Direct Net, and it is simply referred to as a base mesh.

2) \( M \) is the label set of the current Place, and it is used to identify the number of Tokens in each Place of Place sets \( P \). According to the state transformation \( \{M_2, M_3, M_2, \ldots\} \), \( M_0 \) is the initial label. To the current \( p \in P \), if \( M(p) = k \), then the current Place \( p \) has \( k \) Tokens.

3) The rules of Transition are as follows:
   ① To the Transition \( t \in T \), if \( \forall p \in P : p \in t \rightarrow M(p) \geq 1 \), the Transition \( t \) has happened power under the label \( M \), which is marked for \( M[t] > \).
   ② If \( M[t] > \), the Transition \( t \) is happened under the label \( M \) and can get a new label set \( M' \), which is marked for \( M[t] > M' \). To \( \forall p \in P \),
   \[
   M'(p) = \begin{cases} 
   M(p) - 1, & \text{if } p \in \{t\rightarrow\} \\
   M(p) + 1, & \text{if } p \in \{t\leftarrow\} \\
   M(p), & \text{others}
   \end{cases}
   \]

From the definition of Petri Net, all of its possible operation is completely determined by the base network \((P, T, F)\) and the initial lable \( M_0 \).

3.3 PNML

PNML is a description language based on XML of Petri Net. The legitimacy of the label and value in PNML is defined by PNTD (Petri Net Type Definition) [29]. PNML also describes the Figure information, for example Figure location. However, the graph can be independent from the Figure description.

The structure of PNML consists of six parts [29], meta model, type definition interface, feature definition interface, practice document, Petri Net type definition and PNML file.

3.4 CPN

CPN is an extension of Petri Net [30]. In CPN, the data values is attached to the Token of each Place, which therefore forms a Token Color. Token Color of all the Places together constitute the Token Color set; Arc expression on connecting arc determines that how many Token Color will be transferred from the input Place to the output Place [30-31].

Definition 4: CPN [31]

CPN can be formally defined as an eight tuples.
\[ CPN = (P, T, F, C, W_{p\rightarrow t}, W_{t\rightarrow p}, M, M_0) \]

1) \( (P, T, F) \) is a Direct Net and is simply referred to as a base mesh.

2) \( C \) is a Token Color set, \( C = \{c_1, c_2, \ldots, c_k\} \).

3) Both \( W_{p\rightarrow t} \) and \( W_{t\rightarrow p} \) are the function on \((P \times T) \cup (T \times P)\), the fronter is the function on the connecting arc from Place to Transition, the latter is the function on the connecting arc from Transition to Place.
\[
\begin{align*}
W_{p\rightarrow t} &= W(p, t) \in F \rightarrow L(C)_+ \\
W_{t\rightarrow p} &= W(t, p) \in F \rightarrow L(C)_+ \\
M: P \rightarrow L(C)
\end{align*}
\]

Among them, \( L(C)_+ \) is defined as nonnegative integer linear function in Token Color set; \( L(C)_+ \) is defined as \( L(C) \) in which all coefficient is not zero. As shown below,
\[
\begin{align*}
L(C) &= a_1 c_1 + a_2 c_2 + \cdots + a_k c_k \\
L(C)_+ &= b_1 c_1 + b_2 c_2 + \cdots + b_k c_k
\end{align*}
\]

4) The rules of Transitions are as follows.
① for the Transition $t \in T$, if $\forall p \in P$: $p \in 't \rightarrow M(p) \geq W_{st}$, the Transition $t$ has happened power under the label $M$, which is marked for $M[t]$.

② if $M[t] >$, the Transition $t$ is happened under the label $M$ and can get a new label set $M'$, which is marked for $M[t] > M'$. To $\forall p \in P$,

$$M'(p) = \begin{cases} M(p) - 1, & \text{if } p \in 't - t' \\ M(p) + 1, & \text{if } p \in t' - 't \\ M(p), & \text{others} \end{cases}$$

4. CPN of OWL-S Service

The formal definitions for CPN of OWL-S atomic service and CPN of OWL-S combination service are given below. Using a hierarchical description, CPN of OWL-S combination service has been improved. According to the definitions given, CPN of OWL-S combination service can be decomposed to the final CPN of OWL-S atomic service.

4.1 Petri Net of OWL-S Atomic Service

**Definition 5: CPN of OWL-S Atomic Service [32]**

CPN of OWL-S atomic service can be formal defined as a five tuples, corresponding to the graphical representation shown in Figure 3.

$S = (P, T, F, C, Exp)$

1) $P = \{\text{in}, \text{out}\}$, in represents the input Place of atomic service, out represents the output Place of atomic service.

2) $T = \{\text{service}\}$, it represents the Transition of atomic service.

3) $F = \{(\text{in}, \text{service}), (\text{service}, \text{out})\}$, it represents the arc set of atomic service.

4) $C = \{(\text{Input}(S), \text{string}); (\text{Output}(S), \text{string}); \text{Precons}, \text{Bool expression}; \text{EffCons}, \text{Bool expression}\}$

non-empty finite set, $C(p)$ represents the value of Token Color in a Place.

Input($S$) = Input($s_1$)\slash Input($s_2$)…\slash Input($s_n$), $n \geq 1$

Output($S$) = Output($s_1$)\slash Output($s_2$)…\slash Output($s_m$), $m \geq 1$

Precons represents the boolean expression for prerequisite. EffCons represents the boolean expression for output effect.

5) $\text{Exp} = \{\text{E(in \rightarrow service)}, \text{E(service \rightarrow out)}\}$, it is arc function set, in which,

E(in \rightarrow service) = Input(S) \land Precons ; E(service \rightarrow out) = Output(S) \land EffCons.

4.2 OWL-S Process Model

OWL-S process model gives Sequence, Split, Split-Join, Any-Order, Choice, If-Then-Else, Repeat-While, Repeat-Until to define the various components of the combination service [33], as shown in Figure 4. Various specific process model structures are shown in Figure 5, the black box is using for connecting service operation.
The Composite Of Sequence, Split, Split-Join, Any-Order, Choice, If-Then-Else, Repeat-While, Repeat-Until Structures And Services

Figure 4. The Process Model of OWL-S used to Define the Combination Service

1) Sequence Structure, the different services are executed in descending order.
2) Split Structure, concurrent execute a set of services.

Figure 5. Eight Kinds of Process Model of OWL-S Process
3) Split-Join Structure, base on Split, conditions t2 is trigged when all of the parallel Web Services are completed, in which, t2 represents Join effect.
4) Choice Structure, select a service in a group of services to execute.
5) If-Then-Else Structure, Service1 will be executed after satisfying the conditional expression [p], otherwise Service2 will be executed.
6) Any-Order Structure, Service1 and Service2 will be executed with any order, but cannot be executed simultaneously.
7) Repeat-While Structure, execute loop service, first determine by the conditional expression [p], service will be executed if expression [p] is true, or exit the loop.
8) Repeat-Until Structure, execute loop service, the service will be executed first, then determine by the conditional expression [p], service will continue to be executed if expression [p] is false, or exit the loop.

4.3 CPN of OWL-S Combination Service

**Definition 6: CPN of OWL-S Combination Service**

OWL-S compound service of CPN can be formal defined as a six tuples:

\[ SF = (P, T, F, C, Exp, Proc) \]

1) \( P = \{\text{in}, \text{out}\} \) represents the input Place of atomic service, out represents the output Place of atomic service.
2) \( T = \{\text{Service}\} \) represents the Transition of atomic service, and it can be only one and replaced by Stru. Stru is a collection of structures including combination service and atomic service.
3) \( F = \{\text{in}, \text{Service}\}, \{\text{Service}, \text{out}\}\) represents the arc set of combination service.
4) \( C = \{\text{Input}(S), \text{dataType}; \text{Output}(S), \text{dataType}; \text{Precons}, \text{Boolean expression}; \text{EffCons}, \text{Boolean expression}\} \) represents Token Color and the data type of non-empty finite set, \( C(p) \) represents the value of Token Color in a Place.

\[ \text{Input}(S) = \text{Input}(s_1) \land \ldots \land \text{Input}(s_n), \quad n \geq 1 \]
\[ \text{Output}(S) = \text{Output}(s_1) \land \ldots \land \text{Output}(s_m), \quad m \geq 1 \]

\( \text{Precons} \) represents the boolean expression for prerequisite; \( \text{EffCons} \) represent the boolean expression for output effect.
5) \( \text{Exp} = \{E(\text{in} \rightarrow \text{Service}), E(\text{Service} \rightarrow \text{out})\} \) represents the arc function set, in which, \( E(\text{in} \rightarrow \text{Service}) = \text{Input}(S) \land \text{Precons} \land E(\text{Service} \rightarrow \text{out}) = \text{Output}(S) \land \text{EffCons} \).
6) \( \text{Proc} = \{\text{IS}, \text{AS}, \text{Stru}\} \) represents the process model of the combination service, this process model can be defined further as a three tuples, among them, IS is the process model of the combination service, AS is the process model of the atomic service, Stru is a collection of structures including combination service and atomic service.

\[ \text{Stru} = \{\text{Stru} | \text{Stru} \in \text{Allst}\} \]
\[ \text{Allst} = \{\text{Sequence}, \text{Split}, \text{Split - Join}, \text{Any - Order}, \text{Choice}\} \]
7) Presentation of the process model structure is as follows.

\( \text{Sequence: } \text{Seq}(S) = \{s_1, s_2, \ldots, s_n\} \) represents services \( s_1, s_2, \ldots, s_n \) have the Sequence structure.
\( \text{Split: } \text{Spl}(S) = \{s_1, s_2, \ldots, s_n\} \) represents services \( s_1, s_2, \ldots, s_n \) have the Split structure.
\( \text{Split - Join: } \text{Sjo}(S) = \{s_1, s_2, \ldots, s_n\} \) represents services \( s_1, s_2, \ldots, s_n \) have the Split - Join structure.
\( \text{Any - Order: } \text{Ano}(S) = \{s_1, s_2, \ldots, s_n\} \) represents services \( s_1, s_2, \ldots, s_n \) have the Any - Order structure.
Choice: Cho(S, E), S = \{s_1, s_2, \ldots, s_n\}, E = \{e_1, e_2, \ldots, e_n\}, it represent services s_1, s_2, \ldots, s_n have the Choice structure, the service s_i is executed if the corresponding conditional expression e_i is true.

If – Then – Else: Ite(s_1, s_2, e_i), it represents services s_1, s_2 have the If – Then – Else structure, e_i represents the conditional expressions for judgment.

Repeat – While: Fwh(s_1, e_i), it represents service s_1 have the Repeat – While structure, e_i represents the conditional expressions for While judgment.

Repeat – Until: Reu(s_1, e_i), it represents service s_1 have the Repeat – Until structure, e_i represents the conditional expressions for Until judgment.

5. Project Design

According to the front mentioned definition, the following consideration in project is how to use CPSWS to concretely describe CPN of OWL-S atomic service, OWL-S combination service, and how to use the combination design of Tuscany and Spring to realize the work train of thesis.

5.1 Overall Design Thought

Project overall design thought is as shown in Figure 6 and explains as follows.

1) Tuscany is considered as the container of atomic services to develop and deploy services. Through the cooperation of the configuration file .composite of Tuscany and the source code, SCA component and composite can be bound for Web Services, which can realize the reference operations between services. Besides, when forming composite, we can adopt three main interaction way, which include synchronous, asynchronous and callback asynchronous between the components.

2) Component and composite Web Service provided by Tuscany are all converted into CPN of OWL-S atomic service by SCAToOWL algorithm and described by CPSWS. The correspondence relationship of Web Services and Petri nets, the formal description of Petri nets and CPN formal description of the OWL-S atomic service are synchronously stored in the database.

3) Project implementation use Spring beans to implement components as far as possible so that we can make full use of the characteristic of the Spring AOP (Aspect Oriented Programming) and IoC (Inversion of Control), and can use the Before Advice of Spring Bean to realize the Petri E(in → Service) and the After Advice to implement Petri E(Service → out).
5.2 Description of CPSWS

PNML structure is complex, and it includes some redundant information, for example, image information. To simply express CPN of OWL-S service, by combining PNML, we put forward to the below design thought of CPSWS.

1) Using XML file to describe CPSWS, and XML Schema file to describe XML syntax constraints.
2) No longer describe the image information because we can deduce the graphics by the parsing to XML.

3) According to the formalized definition of CPN of OWL-S service, we redefine the XML structure.

4) Increase QoS description as an option, which is mainly used to describe the service response time, the cost of the using service, service reliability, service availability, service security, etc. In this article, we do not focus on the research of QoS.

Based on the above ideas, the design of CPSWS structure is as shown in Figure 8.

The underside of <net> tag includes <serviceType>, <placeSet>, <transitionSet>, <expressionSet>, <the processes> and <QoSSet>, which respectively stand for service types, Place collection, Transition collection, expression collection, process model and QoS collection (optional). <in> in the <placeSet> represents service input parameters, and <out> stands for service output parameters. <atomicService> in the <transitionSet> stands for atomic service, and the <combinationService> stands for compound service. <inArc> and <outArc> in the <expressionSet> respectively stands for import Place and output Place. <processes> process model includes <atomicServiceSet> atomic service collection, <combinationServiceSet> compound service sets and <structSet> structure.
collection. <atomicServiceId> stands for the Id value of atomic service, whose concrete description of the Id values parallelism can be done in other <net> tags; <combinationServiceId> stands for the Id value of combination service, whose concrete description combination service of the Id value parallelism is available in other <net> tags. <structType> in <struct> stands for structure type, Seq, Spl, etc.; However, <serviceId> points to the Id value of participating services, which can be atomic services and combination services.

CPN of OWL-S service is hierarchical and recursive, it is as shown in the tree structure of XML Schema, can use a tree data structure to describe.

5.3 Description of SCA Component and Composite

Component is the foundation part of the SCA assembly process, and it is used to represent a specific business function [25, 26]. Multiple components can be used and configured as the same implementation, and each component can additionally has different configuration for implementation [35].

Component is tated as child element of composite in .composite file. The .composite file is actually an XML configuration file, in which <component> tag is used to represent a component. <component> label is the child element of <composite> label, and <composite> label is on behalf of the assembly components. A <composite> tag can have zero or more <component> labels.

The structure of <component> label is as shown in Figure 9(1). name attribute is necessary in <component> label, and it is used to point out the name of the component. Name of all the components in the same composite must be unique. The autowire property, whose default value is false and optional, and it pointed out whether the contained component reference is automatically connected.

<component> label has zero or one child lables <implementation> which is used to specify usage implementation of components. The components of no <implementation> child labels can not be run. However, perhaps during the development process of a top-down mode and before implementation was developed, the component is useful as a specific needs means of a defined implementation.

<component> label has zero or more <service> labels which are used to configure component services. A configurable service is defined by the implementation.

<component> label has one or more <reference> child labels which is used to configure component reference. The reference that you can configure is defined by the implementation.
Figure 9. XML Description of Component and Composite

<component> label has zero or more child labels <property> which are used to configure the property data values of the implementation.

SCA composite includes a series of components, services and references, the connection wire within them connected each other and a series of attributes used to configure components. Composite can also be as the form of component implementation in the higher composite [25, 26]. The assembly diagram of SCA composite is as shown in Figure 10.

Figure 10. SCA Composite Assembly Diagram

<composite> label structure is as shown in Figure 9(2). Name attribute of <composite> label is used to point out the name of the composite. The name is actually an XML QName, and it is in the namespace of targetNamespace attribute identification. The TargetNamespace attribute is required, and it is pointed out the namespace of combined members. The autowire property is optional, it is a boolean value that default value is false, and it points out whether the contained component reference is automatically connected.
Composite service defined the public service provided by the composite and could be visited by the external combined members. Assembly references description has service description information, which is provided by the composite depending on its external. Wired line expresses the connection between the component service of composite and the component reference of composite.

Composite service is one service promotion of one component in the assembly, and it is actually provided by one internal component. The composite reference is one or more reference promotion of one or more components. As long as all the component references are compatible with each other, the multiple component references can be promoted to the same composite reference. All share the same configuration including the same goal service.

5.4 SCAToOWL Algorithm

SCAToOWL Algorithm, the Algorithm is used to Converted to CPSWS File by Parsing Composite File

```java
// Get all composite files
CompositeFile[] compositeFiles[] = getAllCompositeFile()
// Traverse .composite files
For i=0 To CompositeCount Step 1
    // Find <binding, Ws> label
    WebServiceArray[] wsArray = FindBingWS(CompositeFiles[i])
    // Traverse Web Service array
    For j=0 To wsArray.length Step 1
        // If the element has recorded into the database, then exit
        If wsArray[j].in in DataBase Then ExitFor
        PertriOWL-SNet net = new PertriOWL-SNet()
        // Set the type of service network as atomic type
        net.serviceType = "atomic"
        // If it is binded Web Service type to component
        If (wsArray[j].wsLabel in ComponentLabel) Then
            // Generate network Id
            net.id = wsArray[j].componentId
            // Generate the parameter set of input Place
            net.placeSet.in = wsArray[j].inParamsSet U wsArray[j].propertiesSet
            // Generate the parameter set of output Place
            net.placeSet.out = wsArray[j].outParamsSet U wsArray[j].propertiesSet
            // Generate Transition service invocation parameters
            net.transitionSet.atomicService = wsArray[j].bindingParams
            // Generate network E(in → Service)
            net.expressionSet.inArc = wsArray[j].implementSpringBean.BeforeAdvice
            // Generate network E(out → Service)
            net.expressionSet.outArc = wsArray[j].implementSpringBean.AfterAdvice
        End If
        // If it is binded Web Service type to composite
        If (wsArray[j].wsLabel in CompositeLabel) Then
            // Generate network Id
            net.id = wsArray[j].compositeId
            // Generate the parameter set of input Place
            net.placeSet.in = wsArray[j].promotedService.inParamsSet U
            wsArray[j].promotedProperties.propertiesSet
            // Generate the parameter set of output Place
            net.placeSet.out = wsArray[j].promotedProperties.outParamsSet U
            wsArray[j].promotedService.propertiesSet
            // Generate Transition service invocation parameters
            net.transitionSet.atomicService = wsArray[j].bindingParams
            // Generate network E(in → Service)
```
Known from the algorithm description, when generating the input and output parameters of atomic service, all attributes of the SCA component and composite processes the U operation with the input and output parameters. Because the component attribute values through the Spring to implement can be change, it can be used as the input of a component or a composite, and it also can be used as the output of them. In addition, as the attribute values of composite, it is necessary to see what properties it promotes. When generating the network of the composite is derived from the services promoted. E(out → Service) of the composite is derived from the references promoted.

In addition, whether SCA component or composite, they are both converted into atomic services in the algorithm because OWL-S does not express the SCA internal assembly condition, such as synchronous, asynchronous invocation style. Composite can only be considered as a whole.

6. Case Analyses

When we developed a computing services process of solver on the cloud platform for industrial design and simulation, the process of computing could be find as follows [34, 35]. Firstly, a user submits computing needs, such as the number of CPU used to compute, the number of License, memory capacity, possible computation time and solver types needing to be provided by cloud computing platform. According to the needs of these users, the cloud computing platform processes pre-deduction to the user's account. Then the input file for computing provided by user is uploaded to the supercomputer through the cloud computing platform. If it is fail for uploading file, we say computing job fail, and return pre-deductions to the user. If it is successful for uploading file, the cloud computing platform will carried out CAE simulation parallel computing operations. If the computing operation is fail, the cloud computing platform return the pre-deductions to user. Otherwise, formally charge back to the user account. After the computing successfully, user can download the results of computing, whether the download is successful or not, the account of user should be billinged because the computing has been successful.

Assuming the user has achieved a series of components and composites by Tuscany. Taking PrePayComposite composite as an example, the case of its SCA composite assembly is shown in Figure 11, according to SCAToOWL algorithm, it can get the definition of CPSWS and CPN of OWL-S atomic services.
The pre-charge JavaBean in Spring container made a Before Advice when achieved PrePayComponent, the Before Advice used to check whether existing idle resources on the supercomputer can meet user needs.

SCA artifacts assembly drawing, PrePayComposite composite promotes services of PrePayComponent component and bound to Web Service. Property User denotes user. Property ResourceRequirements includes the number of CPU, memory capacity, the number of licenses for CAE software, possible computation time, etc. Property InputFiles denotes the input file set. Property Solver denotes CAE solver.

Component PrePayComponent also references component AccountOperComponent and component UserOperComponent. Component AccountOperComponent encapsulates the operations to the accounts of users. Component UserOperComponent encapsulates the operations to users.

The method for performing the pre-deduction is as follows.

Public boolean prePay(); //Returns whether the success of the pre-deductions CPN of OWL-S atomic service can be formally defined as a six tuples.

1) \( P = \{ \text{in}, \text{out} \} \)
2) \( T = \{ \text{Service} \} \)
3) \( F = \{ \{\text{in, Service}, \{\text{Service, out}}\} \}
4) \( C = \{ \text{Input}(S), \text{Object}; \)
   \( \text{Output}(S), \text{Object}; \)
   \( \text{Precons}, \text{Bool expression}; \)
   \( \text{EffCons}, \text{Bool expression} \}
5) \( S_{\text{in}} = \{ \text{User, ResourceRequirements, InputFiles, Solver} \}
6) \( S_{\text{out}} = \{ \text{User, ResourceRequirements, InputFiles, Solver, IsPrePaySuccess} \} \),

IsPrePaySuccess showed the success of the advance payment.

5) \( \text{Precons = resourceIsEnough(ResourceRequirements)} \)
6) \( \text{EffCons = IsPrePaySuccess} \)

5) \( \text{Exp = \{E(in \rightarrow Service), E(Service \rightarrow out)\} among them} \)
6) \( \text{E(in \rightarrow Service) = Input(S) \& Precons ; E(Service \rightarrow out) = Output(S) \& EffCons} \)

6) Proc = \( \{\text{IS, AS, Stru} \}, \text{IS = } \emptyset, \text{AS = } \emptyset, \text{Stru = } \emptyset \)
After conversion by SCAToOWL algorithm, automatically combining with OWL-S will be carried on the result of combination is formed a hierarchical CPN shown in Figure 12.

Without loss of generality, we analysis the description for overall service in the first layer and combination service of solving computing in the second layer, Here are the two key definition part of composite services.

1) Service name——SF, which layer——the first layer, description——the description for overall service in the first layer

1) Service type, combination
2) $S_{in} = \{User, ResourceRequirements, InputFiles, Solver\}$
3) $S_{out} = \{User, ResourceRequirements, InputFiles, OutputFiles, Solver\}$
4) $PreCons = true$
5) $EffCons = true$
6) $IS = \{SF_{SACO}\}$
7) $AS = \{SF_{PrePay}, SF_{Pay}, SF_{BackPay}\}$
8) $Stru = \{Seq(SF_{PrePay}, SF_{SACO}, Iter(SF_{Pay}, SF_{BackPay}, IsSolveSuccess)), \}$

$Ite(SF_{Pay}, SF_{BackPay}, IsSolveSuccess)$
2) Service name — SF\textsubscript{SEAD0} — which layer — the second layer, description — the description for combination service of solving computing describes in the second layer

1. Service type, combination.
2. \( S_m = \{ \text{User, ResourceRequirements, InputFiles, Solver, IsPrePaySuccess} \} \)
3. \( S_{out} = \{ \text{OutputFiles, Solver, IsSolveSuccess} \} \)
4. PreCons = IsPrePaySuccess
5. EffCons = IsSolveSuccess
6. IS = \{SF\textsubscript{UpFiles}, SF\textsubscript{COA0}\}
7. AS = \{SF\textsubscript{UpFail}\}
8. Stru = \{Seq(SF\textsubscript{UpFiles}, \text{iie}(SF\textsubscript{COA0}, SF\textsubscript{UpFail}, IsUpSuccess)), \text{iie}(SF\textsubscript{COA0}, SF\textsubscript{UpFail}, IsUpSuccess)\}

This is the interface to submit jobs before the calculation process of job submission as shown in Figure 12 (a). The System accords to the job description of CPN to automatically find each functional stage requires the Web Services after job submission and automatic combine these Web Services. As shown in Figure 12 (b), the interface to monitor jobs can see job execution, such as operation success download state indicate the job has completed but not yet accounting calculation. It also can be seen B/S lets the user interactive download the result files in the interface to monitor jobs.

![Figure 12. The Interfaces to before and after Computing Process](image-url)
7. The Further Work

The next research work can be summarized as following.

1) Study the possibility of whether Tuscany component, composite and CPSWS description files could be integrated into as a whole. Reconstruct or expand the SCA standard XML Schema, so that SCA components, description file of composite member could support OWL-S semantics, Petri Net, automatic service combination.

2) Improve CPN of OWL-S services, and then form new operators and functions, timing trigger functions, optimization calculation, etc.

3) Research the application description of CPN of OWL-S service with related QoS descriptive information and research related QoS algorithm.

4) Research mapping relationship between Tuscany composite and CPN of OWL-S combination service, and the algorithms to convert each other for the component what internal structure is based on a composite as a Web Service reference.

8. Conclusions

CPN has strict formal definition with an intuitive graphical description which can express process structure. OWL-S can be used to support automated combination. In this paper, we form CPN of OWL-S services, and propose CPSWS description language considering the advantage of PNML.

For engineering design aspect, we implement a Web Service by combination Tuscany and Spring. We proposed SCAToOWL algorithm for the demand of converting composite in Tuscany into CPN of OWL-S services. Tuscany and Spring are mature open source software, they are easy to facilitate integration, extensions and programming.

We verified our work from the industrial design simulation process job on cloud computing platform, and our work is easy for project implement. While our work could be further study from many other aspects, such as Tuscany description, integration of CPSWS description, improving CPN of OWL-S services, SCA composite and OWL-S combination service, and mapping relationship between Tuscany composite and CPN of OWL-S service.

The research work in the paper will promote the improvement of PNML to meet the needs of CPN of OWL-S services, and use mature open source software in Engineering Technology (such as Tuscany, Spring) with design idea for CPN of OWL-S services. The research work in the paper will help realize the jobs calculation and other business needs through automatic combining Web Services in the cloud platform for industrial design and simulation.

ACKNOWLEDGEMENTS

This work is supported by the china national science and technology support program, technology development and integration technology platform based on industrial design resource center of the supercomputer (No.2012BAH09B02) and Changsha municipal major science and technology projects, cloud platform and industrialization demonstration application based on the supercomputer (No.K1204006-11-1).

References


Authors

Ziyun DENG is a doctoral student of Hunan University. He is a professor. His recent research interests are in high performance computing, logistics information technology.

Lijun CAI is a professor of Hunan University. His recent research interests are in high performance computing, cloud computing, bioinformatics.

Xiaolin LIANG is a master degree candidate. His recent research interests are in cloud computing, Parallel computing.

Jing ZHANG is a professor of Hunan University. His recent research interest is in control and optimization of complex industrial.

Lei CHEN is a doctoral student of Hunan University. His recent research interests are in cloud computing, big data.