This is the published version (version of record) of:


Available from Deakin Research Online:

http://hdl.handle.net/10536/DRO/DU:30028514

Copyright: ©2009 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.
Facilitating Dynamic Web Service Composition with Fine-granularity Context Management

Wenjia Niu\textsuperscript{1,2}, Gang Li\textsuperscript{3}, Xinghua Yang\textsuperscript{2}, Xu Han\textsuperscript{1,2}, Zhongzhi Shi\textsuperscript{1}

\textsuperscript{1}Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences, 100190, Beijing, China
\textsuperscript{2}Graduate School of the Chinese Academy of Sciences, 100039, Beijing, China
\textsuperscript{3}School of Information Technology, Deakin University, VIC 3125, Australia

{niuwenjia, yangxh, hanxu, shizz}@ics.ict.ac.cn
\textsuperscript{gang.li}@deakin.edu.au

Abstract

Context is an important factor for the success of dynamic service composition. Although many context-based AI or workflow approaches have been proposed to support dynamic service composition, there is still an unaddressed issue of the support of fine-granularity context management. In this paper, we propose a granularity-based context model together with an approach to supporting the intelligent context-aware service composing problem. The corresponding case study is provided to show the validity of our approach.

Keywords- Context Granularity; Web Services; Dynamic Composition; DDL

1. Introduction

As one of the central issues identified in the SOA roadmap, Web services are now widely adopted as a means to interconnect networked programs over the Internet. Developed around some platform-independent protocol standards (e.g. SOAP\textsuperscript{1}, WSDL\textsuperscript{2}, and UDDI\textsuperscript{3}), Web services can be easily deployed, located and invoked across the Web. The main challenge of Web service research is how to employ multiple, heterogeneous developed services to realize a new service to meet the user’s requirement, this problem is of high complexity and generally referred to as Web service composition.

Web services can be created and updated in real time, thus the composition process needs to be aware of the real time updating so that it make decisions on the fly. Moreover, different organizations use different concept models to describe Web services, and there is no unique definition and evaluation on the Web services. Therefore, to tackle these problems, lots of approaches have been proposed to try to build dynamic semi-automated or even automated composition systems. Most existing research fall in the realm of cross-enterprise workflow composition or AI planning. On the one hand, the composite service is similar to a workflow in many aspects, thus the current achievements on dynamic workflow (e.g. WS-BPEL\textsuperscript{4}), automatic process adaptation and cross-enterprise integration provide the means to bind the abstract nodes with the concrete services for automated Web services composition. On the other hand, each Web service can be specified by its preconditions and effects, and actually alter the states of the world after its execution. Therefore when the preconditions and effects of the composite service are specified, a plan can be generated automatically by AI planners through logical reasoning (e.g. OWL\textsuperscript{5}, OWL-S\textsuperscript{6}, SHOP2[1]).

The aforementioned approaches of dynamic composition initially fall along the line of research such as AI planning or workflow, and gradually some individual and concrete descriptions for characterizing different aspects of Web services are considered to affect the composition. These descriptions are generally referred to as “context”, which has been given different definitions by different researchers since the term “context-aware computing” was first introduced in computer science by Schilit in 1994. Those definitions, all have an common understanding that context is any aspect of information of current situation for an entity (e.g. preference, activity, location, time etc.). In addition to the binding of service components, contexts can be used to adjust the composition or execution to provide the client with a customized and personalized value-added service. For instance, a visitor’s hotel preference includes sea-view, as a result, among a number of hotel reservation services, those hotels which are very far from the beach are not appropriate, and on the contrary those close to the beach will be more suitable.

Typically, SHOP2 integrates the context and semantic information during a service composition, in which parameters are considered as contexts information such as the execution environment. In the workflow-based e-service system MAIS[2], the context manager is accessed both by service providers and service consumers and

\textsuperscript{1} http://www.w3.org/TR/soap/
\textsuperscript{2} http://www.w3.org/TR/wsdl/
\textsuperscript{3} http://uddi.xml.org/
\textsuperscript{4} http://www.oasis-open.org/committees/wsbpel/
\textsuperscript{5} http://www.w3.org/2004/OWL/
\textsuperscript{6} http://www.w3.org/Submission/OWL-S/
linked to an interaction platform via an adaptive channel, and finally assists MAIS performing the composition based on the adaptation rules, which encode service negotiation and allow the dynamic adaptation of the execution flow of the services. Another industrial framework is also proposed in [3], in which the context information is passed within a SOAP header between Web Services and during execution a Web Service may change its context information by inserting different context information into the SOAP header and sending the SOAP message to another Web Service. Other typical context-based Web service composition systems, such as GLWSA[4], MOEM[5], COCOA[6] etc., also successfully incorporate the context into Web service composition to some extent. However, none of them considers the influence of context granularity, which exists in real application scenarios of dynamic service composition. Take a composition instance in Google Maps for example (as Fig.1 shown). There are two composition tasks, and one is to find the public transportation from place B (Beijing Guanghua Road) to place A (Beijing Zhichun Road) and the other is to find the public transportation on the opposite direction (from A to B). For the first task, the system composes a bus service and a subway service and tells the user transfer to subway line 10 after taking the No.126 bus. While for the second task, the system still composes the above two services and tells the user firstly take subway line 10 and then transfer to No.126 bus, and this usually is difficult for users because the system doesn’t specify which subway exit should be chosen to find the No.126 bus after getting off the subway line 10, and the wrong subway exit will make it difficult to find the pre-planned bus. In general, the subway service is more complex than the bus service according to their inputs and outputs, so we might assign a complex value (see Def.4.1) $C_S$ to the subway service and $C_B$ to the bus service and $C_S$ is bigger than $C_B$. Then in the first task, the composition context can be simply generalized as “bus service seq subway service, $C_B < C_S$”. While in the second task, the context may be more fine-granularity which can be characterized as “subway service seq bus service, $C_S > C_B$, output data of subway service is (subway station, exit) and input data of bus service is (exit, bus station)”, because the composition is from the subway service to the bus service and supposes that it satisfies a pre-defined rule.

As indicated by this example, what brings the composition problem is substantially the context influence rather than service functionality influence. So, ignoring the influence of different context granularity will make the composition lack flexibility and intelligence. Hence, building a context granularity-based intelligent composition approach is an important and also challenging task, and three issues should be concerned.

1) Local context characterization. In Web services, three roles i.e. provider, broker and user, participate in the composition, in which service broker discoveries and composes provider’s services to meet user’s request. From this point of view, to effectively utilize each role’s context for intelligent service composition, an operational description model of context should be studied for deeply characterizing both static and dynamic information of each local context.

2) Global context characterization. To reflect the global context change upon each local context, the relations to bridge different local contexts should be fully revealed for global context characterization.

3) Context granularity management. For the purpose of flexible and dynamic service composition, context can be modeled as a multi-granularity structure with fine management.

Considering these three issues, we propose a multi-granularity context model to model the local context and global context of Web service composition, and further present a fine-granularity context management approach to strengthening the flexibility and intelligence of dynamic service composition.

The rest of this article is organized as follows. Section 2 reviews literature on contexts of Web Services. Section 3 presents a multi-granularity context model based on local and global context characterization. Section 4 in detail explains the context granularity-based Web service composition approach. Section 5 concludes the article and gives some directions for future work.

2. Contexts of Web Services

For the operational use and not suffering from generality and incompleteness of context, Zimmermann proposed an operational definition of context[7]: “Context is any information that can be used to characterize the situation of an entity. Elements for the description of this context information fall into five categories: individuality, activity, location, time, and relations. The activity predominantly determines the relevancy of context elements in specific situations, and the location and time primarily drive the creation of relations between entities and enable the exchange of context information among entities”.

In Web service, there are three roles: user, provider and broker. Each role has its own attributes as shown in Table 1. To characterize each role’s context information, many formalization approaches consider ontology-based or key-value model. The key-value pairs describe the context by
providing the value of context information (e.g. location information) to an application as an environment variable. In particular, key-value pairs are easy to manage, but lack capabilities for sophisticated structuring. Ontology-based models are particularly suitable for being used in our daily life onto a sophisticated structured data utilizable by computers, such as the contextualized ontology. These efforts successfully formalize context on common information such as time, location, preference and profile etc., however, they lack detailed characterization on action aspects of Web services. For this reason, an action-based context model was proposed in our former work [8],

which is with action support and better serves the specific background of Semantic Web Services. The context is denoted as two tuples: ContextType(Action, Role),Where: Action: a pair (P, E), where, P ∈ E are two finite set of formulas used to describe precondition and effect; Role: the entity with which the context keeps true; ContextType: a name of context type. For examples, a CD-selling service can be characterized by activity context as:

\[ \text{Activity(buyCD}(x,y)\text{, provider}_{1}) \]

\[ \equiv \{\text{customer}(x), \text{cd}(y), \text{instore}(y), \neg \text{bought}(x,y)\} \cup \{\text{bought}(x,y), \neg \text{instore}(y)\} \]

### 3. Multi-granularity Context Model

Context model directly affects the dynamic composition of Web services. First of all, modeling context just is usually simplified as a collection of keywords or terms which lack capabilities for sophisticated structures, and can not effectively exploit the relations of context and the power of semantics. On the other hand, single ontology-based context model lacks detailed characterization on action aspect of Web services. Moreover, our action-based context model in [8] also lacks flexible granularity-based context management. Based on these reasons, in this work we propose a multi-granularity tree-based context model, which includes context granularity structure and context relation model described in the following two parts.

### 3.1 Context Granularity Structure

Suppose that a context role of Web service is a hierarchy structure, which consists of several kinds of context attributes; each context attribute can be further represented by a tree structure as follows.

**Definition 3.1** (Action Tree).

An action context of Web service can be represented by a tree: 

\[ T = \text{Act}(AN, AE) \]

where \[ AN = \{p_1, ..., p_n\} \]

the root node \[ p_i \]

represents the action node, other \[ p_i(i \neq I) \in AN \]

represents the precondition and effect node and formula node; the relation between \[ p_i \] and \[ p_j \]

can be represented by edge \[ (p_i, p_j) \in AE \]. The detailed description of action can be found in [9] and the action tree can be seen as Fig. 2.

<table>
<thead>
<tr>
<th>Context Role</th>
<th>User Context</th>
<th>Provider Context</th>
<th>Broker Context</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attributes</td>
<td>Type</td>
<td>Explanation</td>
<td>Attributes</td>
</tr>
<tr>
<td>Profile</td>
<td>user’s personal information</td>
<td>Profile</td>
<td>provider information</td>
</tr>
<tr>
<td>Preference</td>
<td>user’s preference on the service he wants to get</td>
<td>Time</td>
<td>service provider’s time receiving a request</td>
</tr>
<tr>
<td>Location</td>
<td>user’s location when sending a service request</td>
<td>Location</td>
<td>service provider’s location when receiving a request</td>
</tr>
<tr>
<td>Goal</td>
<td>what the users want to get from services</td>
<td>Activity</td>
<td>function description of service</td>
</tr>
</tbody>
</table>

Table 1. Context of Web Services

**Fig.2 Action Context Tree**

Based on the assumption and the definition on action tree, we give the definition of context granularity structure.

**Definition 3.2** (Context Granularity Structure).

A context role of Web service can be represented as a tree: 

\[ T_P = T(P, E) \]

where \[ P = \{p_1, ..., p_n\} \]

the root node \[ p_i \]

represents the context role, other \[ p_i(i \neq I) \in P \]

defines the attributes of context role and the attributes can be further represented by an action tree, namely \[ p_i = \text{Act}(AN, AE)(i \neq I) \];

the relation between \[ p_i \] and \[ p_j \]

can be represented by edge \[ (p_i, p_j) \in E \]. Hence, the context granularity structure can be finally represented by a multi-granularity tree: 

\[ T_M = T(\text{Act}(AN, AE), E) \]

As shown in Fig.3, in a multi-granularity context structure, role node, attribute node and the action node form a coarse-to-fine granularity structure.

**Fig.3 Multi-granularity Context Structure**

### 3.2 Context Relation Model

In the context environment, the relations among role nodes, attribute nodes and action nodes contain some important semantic information, which can help to bridge
different context information and achieve dynamic and intelligent context management. So, we will elaborate these three kinds of relations respectively.

Suppose that the relation between provider roles is the most important one, because it helps to improve the efficiency in the high-level service finding and composition, while other role relations generally refer to the interaction relation. We exploit the topic-related idea in IR area[10], and believe that each provider has service-related relation with those close-connected providers. For example, if one provider provides ticket service, another one provides hotel service, and they are closely connected from service composition view, and if the third one connects these two providers, then we can infer that the third provider provides some ticket-related and hotel-related service (e.g. post service, food service etc.). Based on this assumption, firstly we give the definition on the role relation.

**Definition 3.3(Role Relation).**

Let $N_5$ be a provider role set and $D_5$ be a provider description set correspondingly. If a new provider node $S_q$ is added, then the role relation between $N_5$ and the nodes $S_i \in N_5$ is:

$$R_5(I,S_i) ((D_5,S_i)) = \Pi(S_i,S_j) \ge C$$

(1)

Where $C$ is a threshold constant and in particular when $P(S_j) \le C$, there is no relation between nodes $S_i$ and node $S_j \in N_5$.

To explain this definition, we take a simple example. Now, $N_5 = \{Ticket Provider, Hotel Provider, Java-Certification Provider\}$ and $D_5 = \{Ticket, Hotel, Java-Certification\}$. According to the definition and former relation dataset and probability calculation, a new provider “Taxi Provider” may have the relation (Taxi,Ticket) with “Ticket Provider” and the relation(Taxi, Hotel) with “Hotel Provider” but not have the relation with “Java-Certification Provider”.

**Definition 3.4(Attribute Relation).**

Context bridging allows us to state that a certain property holds between attributes of two different contexts. The basic notion toward the definition of bridge rules are: a bridge rule from the $i$th role’s attribute to the $j$th role’s attribute is a statement of one of the three following forms:

$$i:C \rightarrow j:E, i:C \leftarrow j:E, i:C \leftrightarrow j:E$$

where $C$ and $E$ represent context attributes of different context roles. For example, one user’s time attribute refers to the Beijing time and one provider’s time attribute also refers to the Beijing time, as a result, we can give their relation as:

$$user:Time \rightarrow provider:Time.$$

**Definition 3.5(Action Relation).**

Actions are formed with the following syntax rule:

$$\Pi, \Pi \rightarrow (P, E) | \phi | \Pi \lor \Pi | \Pi \land \Pi | \Pi$$

where $(P, E)$ is an atom action, $\phi$ is a formula. And there are three kinds relations between actions, form an action relation set{$\lor, \land, \rightarrow$}, respectively named as choice, sequential, and iterated relations. The more explanation can be found in [9].

**4. Context-based Service Composition**

Through the construction of context granularity structure and context relation model, the multi-granularity tree-based context model becomes the foundation of in-depth context analysis and application in Web service, since it can bring the following benefits: (1) it can support the management of context information from coarse to fine, so that we can flexibly control the context influence on Web service composition; (2) the context role level and attribute level handle some high-level context processing and restriction in order to reduce the burden of reasoning in action level and further improve composition intelligence and efficiency.

**4.1 Context Query**

As Fig.4 shown, we give three coarse-to-fine granularity-based queries using ontology-similar query language based on the semantic relations of context.

Among these queries, $?R$ represents the role variable and $?Attr$ represents the attribute variable. The detailed query grammar can be found in the W3C document on RDQL. As a summary, the context query process can be flexibly controlled at a certain granularity level according to the real application scenario, which lays the foundation for the later context qualitative and reasoning.

**4.2 Context Qualitative**

We believe context can be quantized, and should also be meaningful and useful for service composition from the following considerations: Firstly, in our context model, the activity context is responsible for perceiving the service description (e.g. input, output etc.), which embodies the function characteristics of service. Since service can be quantized, the activity context should also be quantized. Secondly, the dynamic of service composition depends on the context influence. A quantized context will be more easy to use for service composition in addition to semantic aspects of context. So we got two assumptions: 1) For activity context itself, a named “complex value” should be calculated for representing the complexity of the function characteristic of an activity context; 2) According to the role relation of

---

an activity context, a named “composing value” should be calculated for representing the reasonable composing value of activity context in the composing process. Based on these assumptions, we give the definition of complex value and composing value as follows.

**Definition 4.1 (Complex Value).**
Let $A_k (|A_k|=k)$ be a formula set of action of the activity context $A$ of provider role and $CD_k (|CD_k|=k)$ be a set of concept type of corresponding formula. Then the “entropy” is leveraged to calculate the complex value $CV$,

$$CV(A) = \sum_i p_i \log p_i$$

(2)

Where $p_i$ is the probability of $CD_i$ appeared in $CD_k$.

**Definition 4.2 (Composing Value).**
Let $RelS (|RelS|=k)$ be a formula set of action of the activity context $A$ of provider role, where keeps the role’s composition history and $R_i$ be the first role and $R_j$ be the second one. Then the composing value (COV) of $R_i$ against $R_j$ can be calculated by the formula

$$COV_i (R_j) = |(R_i, R_j)|/(|R_i|, |R_j|)$$

(3)

Where $|(R_i, R_j)|$ is the probability appeared in RelS.

### 4.3 Context Reasoning
In our context model, logic action is utilized to characterize the action aspects of context. For logic action of dynamic description logic(DDL) in [9], there are four kinds of action reasoning named “realizability”, “executability”, “projection” and “plan”, in which “plan” is most used for context reasoning, hence we only give the “plan” definition in brief.

**Definition 4.3 (Plan Reasoning).**
Let $\psi$ be a formula and $\sum$ be a set of actions. Let $\pi_1, \ldots, \pi_n$ be a sequence of actions with each action coming from $\sum$. Then, with respect to an RBox $D_R$, a TBox $D_T$ and an ActionBox $D_A$, the sequence $\pi_1, \ldots, \pi_n$ is a plan to achieve the goal $\psi$ starting from initial states described by the ABox $D_A$ if (i) the sequence-action “$\pi_1, \ldots, \pi_n$” is executable on states described by $D_A$ and (ii) $\psi$ is a consequence of applying the sequence-action “$\pi_1, \ldots, \pi_n$” on states described by $D_A$.

So, when giving a goal formula $\psi$, it is not difficult to achieve all the context actions in order to build the reasoning space, and the rest task can be totally taken by the DDL reasoner, which actually does the context reasoning.

### 4.4 Dynamic Service Composition
Our context-based dynamic service composition has several distinguishing characteristics that set it apart from other approaches mentioned in Sec.1. First of all, context information distributed around the service composition can be quantized and managed to help provide on-demand Web service or composite service with underlying logical reasoning. Secondly, the reasoning space can be built, filtered and updated based on context in a dynamic way, which increases the efficiency of reasoning and service composition. Thirdly, the context value which includes complex value and composing value can be used to assist the reasoning process and further update the service composition dynamically. Algorithm 1 shows our context-based service composition approach in details.

From the algorithm we can see that, each listener embodies the dynamic characteristic of our approach for service composition. As to the knowledge base of composing history mentioned in the algorithm, how to build it and what is its data structure, which we believe are out of the scope of the paper, are ignored.

Now, let us put attention to the solution of the Google service composing problem in the introduction section using Algorithm 1. Suppose three actions are put into the reasoning space:

- Action_1 = {StartPlace(x), EndPlace(y), Line(z), BusStation(p), ¬has(z,p)};
- Action_2 = {StartPlace(x), EndPlace(y), Line(z), SubwayStation(p), ¬has(z,p)};
- Action_3 = {StartPlace(x), EndPlace(y), Line(z), SubwayStation(p), ¬has(z,p)};
- the $\psi$ is $\text{Person}(m) \land \text{StartPlace}(x) \land \text{EndPlace}(y) \land \text{Line}(z) \land \text{has}(m,z)$.

<table>
<thead>
<tr>
<th>Algorithm 1. Context-based Service Composition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Begin</strong></td>
</tr>
<tr>
<td>Step_1(Listener_1): In order to filtering out unnecessary actions in reasoning process, through context query, only add the action of activity attribute of provider role into DDL reasoning space.</td>
</tr>
<tr>
<td>Step_2: Through context query, extract the action of goal attribute of user role to form the goal formula $\psi$ and put it into DDL reasoning space.</td>
</tr>
<tr>
<td>Step_3: Based on knowledge base of composing history, calculate each context’s complex value and composing value, and add these values into a temporal knowledge base.</td>
</tr>
<tr>
<td>Step_4(Listener_2): After Step 1, Step 2 and Step 3, start DDL action reasoning for service composition.</td>
</tr>
<tr>
<td>Step_5(Listener_3): After Step 4, output the result of “plan” reasoning to user and form a composing history into the knowledge base, then according to its result, through context query find corresponding context information of upper level (role level and attribute level) and further output the service composition result.</td>
</tr>
</tbody>
</table>

**Listener_1**

Condition_1: if new context is added;

```
Do_1 { add the new action of activity attribute of provider role into DDL reasoning space. }
```

Condition_2: if existing context is updated;

```
Do_2 { Through context querying, only update corresponding action. }
```

**Listener_2**

Condition: if in the reasoning process, the complex value of first context is bigger than the second one;

```
Do { set reasoning break point, and try replacing the
```
corresponding actions with similar functionalities depending on both the composing value and pre-defined rules.
}
Listener_3
Condition_1: if reasoning is unsuccessful without Listener_2 done;
  Do_1{ null.}
Condition_2: if reasoning is unsuccessful with Listener_2 done;
  Do_2{ backtrack the break point and do recovery, output log information for expert to modify composing rules.}
End

When in the Step_4, the Listener_2 found the “Action_2 seq Action_1”, where $CV(\text{Action}_2) > CV(\text{Action}_1)$, so after setting a reasoning break point, and our approach tries replacing the corresponding actions Action_2 with Action_3, because the $COV_{\text{Action}}(R_{\text{Action}_3}) > COV_{\text{Action}}(R_{\text{Action}_2})$ is satisfied with one pre-defined rule. Please note that there are lots of composting scenarios and it is difficult to prove whether one rule is suitable for all scenarios, hence this pre-defined rule should be only regarded as this scenario-based rule, which can be described and stored as rule template.

5. Conclusion

In this article, we presented a context granularity-based service composition approach, which involves the following aspects: multi-granularity context modeling; context management which includes context query, context qualitative and context reasoning; dynamic aspects of context-based composition. In our future work, we will aim at the experimental analysis of our approach and evaluations from both theory prove and complex case study.

Acknowledgments
This work was partially supported by the National Natural Science Foundation of China (No. 60775035), the National High-Tech Research and Development Plan of China (No. 2007AA01Z132), and the National Basic Research Program of China (No. 2003CB317004, 2007CB311004).

References