A Framework for Composing Web Resources

Hua Xiao
School of Computing
Queen's University
Kingston, Ontario, Canada
huaxiao@cs.queensu.ca

Bipin Upadhyaya, Ran Tang, Ying Zou
Dept. of Electrical and Computer Engineering
Queen's University
Kingston, Ontario, Canada
{9bu, ran.tang, ying.zou@queensu.ca}

Joanna Ng, Alex Lau
IBM Toronto Lab
Markham, Ontario, Canada
{jwng, alexlau}
@ca.ibm.com

Abstract

Large amount of heterogeneous Web resources, such as SOAP-based Web Service and RESTful service exist on the Internet. It is labor-intensive and inefficient for an end-user to search and compose different Web resources in order to fulfill his/her requirement. To support the end-user's various activities, we propose a Web resources composition framework. This framework can help end-user: 1) discover available Web resources to fulfill the end-user's goal, despite of their types; 2) represent the relation between different resources to allow them to be used collaboratively; 3) automatically compose required Web resources to fulfill the goal specified by the end-user.

1 Introduction

Various types of Web resources, such as SOAP-based Web Service and RESTful service, exist on the Web to provide various functionalities, such as information access and online banking. There are large amount of Web resources available on the Internet. Not all Web resources are highly relevant to a user's requirement. Only a subset of Web resources can fulfill an end-user's requirement. It is difficult for the Web end-users to sift through the sheer volume of Web resources to fulfill their goals. Without the aid from a service composition tooling, an end-user has to manually discover and compose different Web resources. For example, a person planning a conference trip needs to locate the Web resources for transportation, accommodation and other activities separately and integrate the results from these Web resources. This is a time-consuming and tedious process and may not produce the optimal outcome. For example, the end-user may not be able to discover the Web resource that provides the most economical air ticket.

To support the end-user's social, professional, recreational and other activities, it is essential to create a Web-based service composition framework that can 1) discover all available Web resources to fulfill the end-user's goal, despite of their types; 2) represent the relation between different resources to allow them to be used collaboratively; 3) automatically compose required Web resources to fulfill the goal specified by the end-user. It is challenging to realize such a service composition framework. In particular, The Web resources are described in heterogeneous formats. For example, Web Service Description Language (WSDL) [2] is used to describe SOAP (Simple Object Access Protocol) based Web Services that makes remote procedure calls. HTTP-based APIs are simpler Web resources, implemented as a set of standard HTTP requests. Examples of HTTP-based APIs are twitter, Flickr and various Yahoo APIs. The HTTP-based APIs can be described using Web pages or WSDL 2.0. Informational Websites are implemented by various technologies, such as Ajax, HTML and XML. Some of the descriptions, such as HTTP-based APIs are not machine readable. This hinders the ability to discover the most suitable resource to satisfy the goal of a particular end-user.

In this paper, we propose a framework to help end-users compose various Web resources. Our framework uses a unified description schema to describe the heterogeneous Web resources and a resource graph model to represent the relations among different Web resources. Representati tional State Transfer (REST) [9] is an architectural style for network-based systems. REST was not introduced as an approach to designing Web services, yet the non-corporate Web Service community as alternative to SOAP/WSDL has adopted it. Although not always adhering to the all of REST's constraints [10], RESTful Web Services are gaining popularity and are adopted

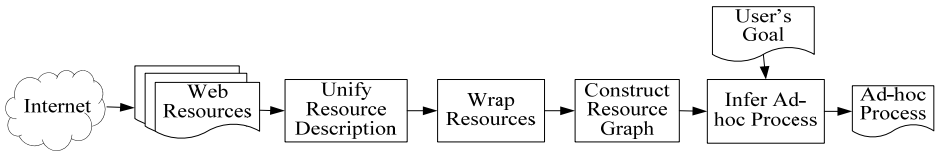

Figure 1: Steps for composing Web resource

by major service providers like Google, Amazon and Yahoo. RESTful service provides uniform interface which is immutable (no problem of breaking clients). HTTP/POX is ubiquitous (goes through firewalls). Since it adheres to the principle of Web, it naturally has proven scalability with caching, clustered server farms for Quality of Service (QoS). End user just need browser to get started, no need to buy WS-* middleware. Moreover, we provide an approach to compose resources for end-users using the resource graph. The composite resources are represented as ad-hoc processes. An ad-hoc process contains a set of tasks without strict execution order.

The remainder of this paper is presented as follows. Section 2 gives an overview of the proposed framework. The details of the unified description schema, the resource graph to model the relation among resources, and the technique to construct ad-hoc process using the resource graph are presented in section 2. Section 3 concludes this paper.

2 Overview of Framework

Figure 1 provides an overview of our framework. To describe the heterogeneous Web resources, we collect heterogeneous Web resources from the Internet and represent them in a unified resource description schema. We identify the relations among different resources and construct a resource graph. HTTP-based APIs are generally described using plain, unstructured HTML documents which are only useful to human developers. Nowadays, using Web resources, such as finding suitable services, composing services, mediating between different data formats, are mainly manual tasks. To maximize the interoperability among the resources, we need a common data model to describe the resources and their relations. There are two requirements for interoperability: (1) the Web resources themselves must be able to programmatically interoperate. For example, they must be able to invoke one another and pass data among themselves; (2) there must be a user interface mechanism for the user to orchestrate the Web resources to work together toward some complex goal. End-users should be able to compose and define the flow between the Web resources. RDF is designed specifically for exchanging and integrating Web data. In our framework, we wrap the unified Web resources into RESTful services then adopt Description Framework (RDF) [1] to describe RESTful services and their relations. While we construct the resource graph, the unified resource descriptions are used to help us identify resources and their relations.

When an end-user wants to fulfill a goal, the end-user simply describes the desired goal using keywords. We map the keywords into resources described by the resource graph. and infer an ad-hoc process from the resource graph to help the end-user fulfill the goal. In the following subsections, we discuss the details in unifying the description of various types of resources, constructing a resource graph, and inferring ad-hoc processes from the resource graph.

2.1 Unifying Resource Description

To assist the automatic discovery of various Web resources, we propose a schema to uniformly describe different types of Web resources. The unified representation provides a better chance to discover Web resources than limiting the service discovery within the Web resources of a single type.

As shown in Figure 2, we define two parts in the unified schema: the general description part and the operation description part.

The general description of a Web resource provides a bibliographic description about the Web resource: the type, the name, the provider and the URI of the Web resource. Such descriptions are common to all types of Web resources. There are standards suitable for representing the general description. For example, using 15 text fields (e.g., title, type and publisher), the Dublin Core metadata schema can describe various resources, e.g., books and Web pages [3]. We adopt the Dublin Core format to represent the general description.

Among the four fields of general description, the type, the name, the provider fields are self-descriptive. The URI field of the general description refers to the URI that identifies the Web resource on the Internet. The URI of a SOAP-based Web Service points to its WSDL file. For an HTTP-based API, the URI field is filled by the URI of its description Web page. To invoke a particular functionality of the Web resource, another URI may be required because each operation of the Web resource may have a different URI, which is described in operation description part.

The operation description describes the functionalities offered by a Web resource. A Web resource can deliver one or more functionalities. An operation represents a primitive unit of functionality used to compose a service-oriented application. Different types of Web resources contain varied number of operations. Most SOAP-based Web Services and HTTP-based APIs provide complex functionalities and contain multiple operations.

To describe each operation, we use the tag-based description, the formal interface and the excerpt of existing description. The tag-based description uses a set of descriptive tags (i.e., keywords) to informally represent an operation, including the functionality description, the input description and the output description. The input/output descriptions help describe different operations with the same name or similar functionality description. For example, two operations are named as "displayOrders". One operation with the parameter "productID" is different from the other one with the parameter "custormerID". The tag-based description can concisely convey the functionality of the operation to the resource consumers. In addition, it facilitates automatically compare the functionalities of operations in order to discover similar Web resources.

The formal interface provides information to support the invocation of an operation. The formal interface is intended for machine consumption in order to facilitate automatic invocation. The SOAP-based Web Service is originally described with a formal interface. Hence, a client program can be automatically generated to invoke operations in a SOAP-based Web Service. In contrast, other Web resources (e.g., the HTTP-based APIs) do not have a formal interface and the SOA professionals need to manually write the request to invoke the operation. The fields contained in the formal interface depend on the type of Web resources. To invoke an HTTP-based API operation and a Web form, the URI, the HTTP verb, and the input parameters of the operation are required. The formal interface is described in an XML format which can be interpreted by a machine to automatically invoke the operation.

An excerpt is taken from the existing description of an operation. It provides more readable and detailed information, such as examples and demonstrations. It also offers a shortcut for a SOA professional to understand the operation without having to search for the operation in the entire document. The excerpt is available only for SOAP-based Web Services and HTTP-based APIs.

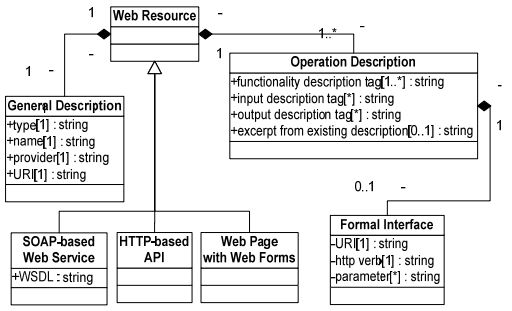

Figure 2: A unified resource description scheme

2.2 Constructing Resource Graph

A resource graph represents all resources using RDF model. It is a semantic network model, which consists of entities and relationships. Entities are identified globally with URIs. We want to

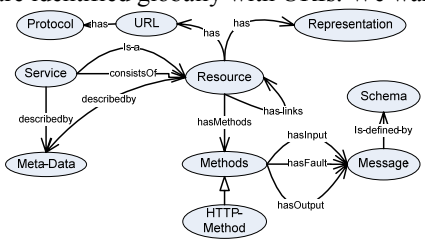

Figure 3:Conceptual model for RESTful Services

represent all the service in REST style. RESTful Service and RDF both represent "Resource," which is the main motivation behind using RDF. RDF provides a common framework for expressing information so it can be exchanged between applications without loss of meaning.

Figure 3 shows the conceptual model of the RESTful services. We model all SOAP based Web Services, HTTP-based API in terms of

RESTful services. As shown in figure 3, each service has one or more resource and service itself can be treated as resource. Each resource is uniquely identifiable and can have one or more than one representation. The information about the service is stored in the meta-data and resources have links to other resources. The input and output message has one or more parameter which is defined by the schema. Resource may link to other resources. We used link specification [7] to indicate the relationship between the resources. The "rel" attribute in the link specification give the information about the semantics of the link. In addition to types of "rel" defined by IANA [8], we introduced few other types that helps to represent resource in RESTful services more easily.

- **see-also** recommends another service
- **same–as** provides the similar services
- **is-a** defines is-a relation between the resources
- **contains** defines different service as in the case of composite service.
- **is-container-of** defines the resource-to-container relationship.

These relations help recommend services, identify similar services, and define the relationship between the resources. Semantic relationship helps to abstract the resource representation. Figure 4 shows the different resource and the semantic and data-link relations between the different resources. Since double bedroom and single bedroom has is-a relationship with room all the data link and semantic relation from room is carried over to those two different categories of the room. In addition to that, we added "method" attribute in the link. Thus, the user agent can know next possible resources to visit and along with method used to visit that resource. The information provided by the method attributes helps to define the flow. It also tells about the data required by another resource that end-user wants to visit. For example in Figure 4, "review" resource requires the information regarding the resource hotel. Hence, this type of relationship is called data-link relations. The small circle represents which method to can be invoked on resources from the current state. In Figure 4 the user can invoke only GET method in the resource review from the hotel resource. This substantially increases the user agents' capability of discoverability of resource. In our resource graph, the resource from where end-users can start consuming service (a starting point) is defined as initial node. In Figure 4, hotel is represented in different color and it denotes the initial node. When a user requests a service this node is returned and from that node, user can start using the service.

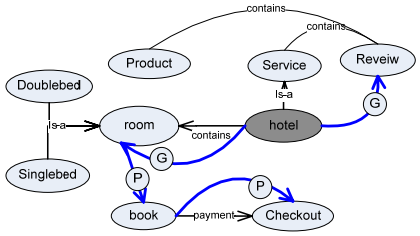

Figure 4: Resource and their relations

For example, if we want to book a room we need information regarding the resource room, Thus using REST approach it can be used as shown in following example:
As shown in Figure 5, when the resource room is requested, the output contains information about the next resource. The next state in this example can be one of the room types which are single bedroom and double bedroom. The semantic relation between these resources is give by "rel" attribute .After knowing the next resource user agent also needs to know which method to use. It is provided in method attribute in the link.

```
GET /hotel
HOST: foo.org
<various HTTP headers>
Output
<rooms>
<room id="singlebed" rate="40" currency="USD">
<link rel="isa" method="GET" href="/rooms/singlebed/">
</room>
<room id="doublebed" rate="40" currency="USD">
<link rel="isa" method="GET" href="/rooms/doublebed/">
</room>
</rooms>
```

Figure 5: Example of RESTful Service

WSDL/SOAP based services are lifted to resource level by identifying the resource and the HTTP method for each operation and then described in RESTful approach. When the end-user uses these resources the corresponding WSDL operations have to be invoked which can be identified using the mapping information for operation and resources. Converting every service in REST style helps the end users to see everything as resource and each resource is associated with the corresponding four methods. End-users are exposed to only the resource model of the service described in RDF. It is easier to deal with one

model than dealing with different kinds of heterogeneous specification. Once the service is represented in the RDF form, it adds a lot of expressiveness with query languages (e.g., SPARQL [4]), transformation languages (e.g., GRDDL [5]), and rule languages (e.g., RIF [6]).

2.3 Inferring Ad-hoc Processes from Resource Graphs

Figure 6 illustrates the definition of ad-hoc processes. An ad-hoc process is characterized by a set of work items and sub ad-hoc processes performed by end-users to fulfill a goal. A work item in the ad-hoc process is a set of tasks which are collaborated together to accomplish a transaction. The work items in an ad-hoc process are connected through the relation defined by users or the semantic relations defined in the resource graph. A task is the combination of resource and operations. The resource in a task is defined in the resource graph. The operation in a task processes the resource. In RESTFul services, we can use operations Get, Post, Put and Delete. For example, the task "search for flight ticket" could be described as the resource "flight ticket" and the associated operation "get". Eventually, a task can be performed by one or several concrete Web services. Web services can be represented as resources in the resource graph. Therefore, we can trace the resource graph to find the associated Web services.

In our definition, work item is different from ad-hoc process although they both contain tasks. The resources related to the tasks in a work item are connected by data-link relations in the resource graph. The relations of tasks in a work item are very closely coupled. To fulfill a transaction, the user needs to execute all the tasks linked by the data. For example, the work item "buy flight tickets" includes tasks "search ticket", "choose ticket" and "pay the bill". In order to accomplish the goal of "buy flight tickets", the user has to perform all the three tasks in the work item. On the contrary, the relations of work items in an ad-hoc are loosely coupled. Different users can have different ad-hoc processes for the same goal. For instance, when planning a trip, some users may prefer taking flight than driving car, but other users may prefer driving instead of taking flight. The ad-hoc process of planning a trip for these two groups of users is different since the ad-hoc process for the former group contains the work item "buy flight tickets" and the latter does not have.

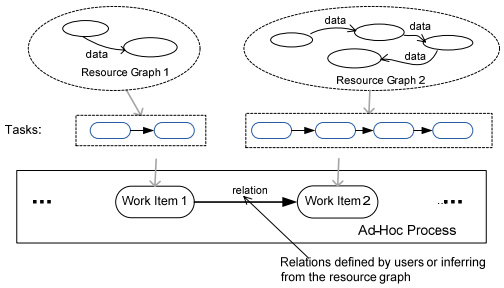

Figure 6: Definition of ad-hoc process

2.4 Inferring Work Item Relations from Resource Graphs

The work items in an ad-hoc process can be connected together using different relations. The relations of work items in ad-hoc processes are shown as follows.

- **Group** indicates that a set of work items should be performed together. The group relation can be further described as "And", "Or", "Sequence" or "Parallel" relations.
- **And** means that all the work items need to be performed. "And" relation does not provide the detailed information about the execution order. Thus the work items can be executed in any order by default. When we have more information about the execution order of work items, we can use "Sequence" and "Parallel" relations to describe "And" relations with the execution order information of the work items.
- **Sequence** means that the work items need to be executed following an order.
- **Parallel** means that the work items can be executed at the same time.
- **Or** indicates that the user only needs to execute one of the work items in the group**.**
- **Ungroup** releases the existing "group" relation. When a user does not like an existing group of work items, the user can use this relation to show that he/she does not agree to group these work items together.

Our framework analyzes the relations in resources graph and infers the work item relations from the resource graph. Table 1 shows the mapping from the resource graph to work item relations.

In Table 1, the "See_also" relation in resource graph can be use to recommend work items to users. The user has the option to perform it or

ignore it. The "Same_as" relation in the resource graph indicates that two resources are equal. Therefore, we convert the "Same_as" relation into "Or" relation of work items. The siblings of "Is_a" relation in the resource graph are converted to "Or" relation since "Is_a" relation shows that one resource is an instance of another and these instances have the same features. The elements in a "Contains" relation in the resource graph is converted to "And" relation in the ad-hoc process.

Figure 7 gives an example to illustrate the main idea of inferring work item relations from the resource graph. In Figure 7, the resources "Hilton", "Holiday_Inn", "Flight", and "Restaurant" are converted to work items in the ad-hoc process. In this example, if we trace entire resource graph, these work items can be implemented by several tasks which are associated with detailed resources.

Table 1: Infer work item relations from resource graphs

Relation in resource graph	Work item relation
A See_also B	A optional B
A See_also B, A See_also C	A optional B, C
A Same_as B	Or (A, B)
A Is_a B, A Is_a C	Or (B, C)
A contains B, A contains C	And (B, C)

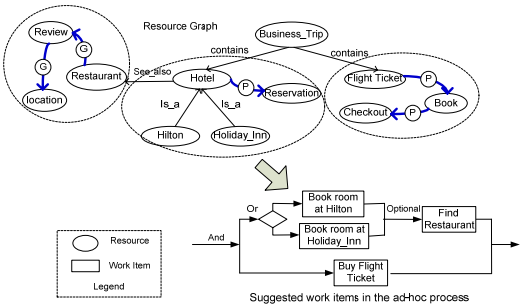

Figure 7: An example of relation inference

3 Conclusion

This paper presents a framework to compose heterogeneous resources on the Web. In our framework, Web resources can be described by the unified description schema and can be wrapped to RESTful services. The resources in our framework have semantic relationship and data link relations between them and can be described in RDF. RDF also adds a lot of expressiveness with query languages, transformation languages, and rule languages bringing more participation from end-users side. Thus we provide a framework whereby Internet end-users can compose their own Internet space, which is defined as a collection of resources that can be used to feed, filter, compose, disseminate, and reference information, data, and services to end-users according their profile, context, and mode of operation. By analyzing the relations among services, we can infer the ad-hoc processes to compose resources. The way of describing everything in terms of resources solves the integrating issues and drives the innovation in the end user side.

References

[1] D. Beckett, B. McBride (editors), "RDF/XML Syntax Specification (Revised)," W3C Recommendation (2004)

[2] R. Chinnici, J. Moreau, A. Ryman, S. Weerawarana, "Web Service Description Language," W3C Recommendation (2007)

[3] Dublin Core Metadata Initiative, http://dublincore.org/, last accessed on October 12, 2010

[4] SPARQL Query Language for RDF, http://www.w3.org/TR/rdf-sparql-query/, last accessed on October 12, 2010

[5] Gleaning Resource Descriptions from Dialects of Languages (GRDDL), http://www.w3.org/2004/01/rdxh/spec, lat accessed on October 12, 2010

[6] RIF In RDF, http://www.w3.org/TR/rif-in-rdf/, last accessed on October 12, 2010

[7] Web Linking, http://tools.ietf.org/html/draft-nottingham-http-link-header-10#section-4, last accessed on October 12, 2010

[8] Link Relations, http://www.iana.org/assignments/link-relations/link-relations.xhtml, last accessed on October 12, 2010

[9] R. Fielding. "Architectural Styles and The Design of Network-based Software Architectures," PhD thesis, University of California, Irvine (2000)

[10] Vinoski, S.: RESTful Web Services Development Checklist. Internet Computing,IEEE 12, 96–95 (2008)