Journal of Universal Computer Science, vol. 19, no. 13 (2013), 1963-1985
submitted: 23/10/12, accepted: 28/6/13, appeared: 1/7/13 © J.UCS

Ontology Combined Structural and Operational Semantics for Resource-Oriented Service Composition

Cheng Xie
(School of Software, Shanghai Jiao Tong University, Shanghai, China
chengxie@sjtu.edu.cn)

Hongming Cai
(School of Software, Shanghai Jiao Tong University, Shanghai, China
hmcai@sjtu.edu.cn)

Lihong Jiang
(School of Software, Shanghai Jiao Tong University, Shanghai, China
jiang-lh@cs.sjtu.edu.cn)

Abstract: Resource-oriented Services recently become an enabling technology to integrate and configure information from different heterogeneous systems so as to meet ever-changing environment which not only need the concepts for entities but also require the semantics for operations. By the aim of combining structural and operational semantics agilely, a Semantic Resource Service Model (SRSM) is proposed. Firstly, SRSM describes Entity-Oriented and Transition-Oriented Resource by semantic meta-model which contains data structures and operation semantics. Secondly, by describing structural semantics for Entity-Oriented Resource, heterogonous inputs/outputs of a service can be automatically matched. Thirdly, by describing operational semantics for Transition-Oriented Resource, the service composition sequence can be inferred after ontology reasoning. Then, both Entity-Oriented and Transition-Oriented Resources are encapsulated into composited RESTful service. At last, a case study and several comparisons are applied in a prototype system. The result shows that the proposed approach provides a flexible way for resource-oriented service composition.

Keywords: Structural Semantic, Operational Semantic, Ontology, RESTful Service, Resource-Oriented Architecture, Entity-oriented Resource, Transition-oriented Resource
Categories: D.2.11, H.1.1, H.4.0

1 Introduction

Resource-oriented services [Pérez et al., 10] composition has been a subject of interest of service composition and configuration in the past ten years. In order to adapt to the rapidly developing and intensively competitive market, enterprises have to adjust their business strategies frequently and change their supported information infrastructures on demands. In some information-centric applications such as ERP system, E-business system, Resource as a Service (RaaS) [Tao et al., 10] emerges as a new trend for exchanging data between independent data providers and data users across heterogeneous and distributing platforms. It is usually a user-interactive and iterative process to compose the services into various data-driven business scenarios of data acquisition, analysis, and other processing activities. Instead of activity series, the execution results are the main concerns for enterprises decision-makers in these situations. Therefore, resource-oriented service composition is important for users to accomplish rapid configuration according to different industrial or business purposes.

However, there exist two main challenges making resource-oriented services composition a difficult task. First, entities are difficult to describe. Business elements of enterprise are always very complex, since they often involve lots of entities such as people, organizations, items and products with the complicated relationships. It is not an easy task to describe and define the structure of these business entities in a non-semantic way. Second, the semantics of operation are difficult to describe [Liu, 11]. Operational semantics which focus on the change of data and relationships are quite different from structural semantics which only concern the definition of concepts. Furthermore, current semantic technology is usually used to describe the static semantics for structure and relation of concepts. And it is difficult to describe an operation such as task, action and process, which is important for information system.

Therefore, a semantic resource service model (SRSM) is proposed to combine structural and Transitional semantics for resource-oriented service composition. Correspondingly, based on SRSM, a supported platform for semantic resource modelling including entity-oriented resource and transition-oriented resource is built to implement resource-oriented service composition.

This paper is organized as follows. Section 2 reviews previous works on ontology-based information system design and ontology modelling. Section 3 introduces the approach of structural and operational semantics modelling based on SRSM. Section 4 presents the reasoning and composition method for Entity-oriented and Transition-Oriented Resource. Then, Section 5 provides a case study on a prototype system. Finally, section 5 concludes the paper and discusses future works.

2 Related works

This section presents an overview of the related works done in the area of the operational semantic representation, entity semantic modelling and service composition.

(1) Ontology-based Entity Semantic modelling

Entities are abstract concepts, and might be considered as a container that holds all of the instances of a particular thing in an information system. In the paper, we focus on the entity semantics modelling in conceptual level based on ontology.

In the early years, Andrea and Egenhofer [Andrea and Egenhofer, 03] present three basic components for the representation of entity in ontology: 1) a set of synonym words that denotes an entity class, 2) a set of semantic interrelations link these entity together, and 3) a set of distinguishing features that characterize entity. By using those three components, the semantics of an entity can be addressed in names, relations and inner features.

In the last few years, many researchers reach a consensus on entity representation. Waters [Waters et al., 09] models entity as a resource with a simple way to represent and share the resource. The main steps of this approach is divided into three parts: 1) identifying a resource, each resource should have a unique id so that different systems can refer to the same “entity”; 2) Representing a resource, entity should not be represented in XML format but higher-level standards that are based XML. such as

RDF[1] and OWL; 3) Managing the resource, each entity ought to run on Resource-Oriented Architecture (ROA), such as REST architecture that addressing every resource with an unique URI and using HTTP methods (POST, GET, PUT, DELETE) to control the resource state transformation.

Now days, some researchers believe that representing and sharing resources is not enough to meet the ever-changed requirements, many efforts are made to find out potential relationships between resources that beyond resource representation. Bradford [Bradford, 10] presents an approach for generating more fine-grained subdivisions of entity type not just to extract entities into a few basic types, such as person, organization, and location. He mainly uses the technique of latent semantic indexing (LSI) to provide semantic context as an indicator of likely entity subtype. Feiyu et al [Feiyu et al., 12] presents a context-based ontology matching approach to find out the potential relationships between entities (resources, instances and concepts). Their approach combines context-based “String Matching”, “Structure Matching” and “Lexical Matching” to efficiently align the relationships between entities.

(2) Ontology-based Operational Semantic Representation

Different from SOS [Plotkin, 04], the operational semantics we discussed are mainly focused on ontology-based operational semantic description.

Zhao [Zhao and Doshi, 09] divides semantic resource into individual resources and transitional resource. And the transitional resource can be used to represent an operation or a task in the semantic level. But the rules and effects of the operation are described by SWRL [Horrocks et al., 04] language which cannot be inferred by most existing ontology reasoner. Hai [Hai et al., 11] provides an ontology-based operational process representation language called WPML. It enables the representation of the behavioural and functional aspects of a work process. But it mainly focuses on describing the input/output states and the relations between action and function not operation itself. Thus, the operations, described in ontology, are still lack of semantics. Wang [Wang et al., 07] proposes Petri-Net ontology to incorporate the operation and IOPE [Tomoaki and Ning, 07] (Inputs, Outputs, Preconditions, and Effects) semantics for system test case generation. Operations and transitions can be organized by reasoning the outward semantics such as preconditions, effects, input and output. But the approach lacked formal representation of effects, which contains primary semantics of the operation.

(3) Ontology-based Information System Design

In our previous work [Cai et al., 12], take the advantage of conceptual ontology, the gap between business modelling in build-time and system configuration in runtime has seamlessly bridged. By means of ontology, these business elements are transformed into IT service-oriented components such as SOAP services, RESTful services. And then, Referring to Model-View-Controller pattern, these services components are configured in a runtime supported environment. However, the services components in this work are design as service aggregation not service composition. Thus, it is hardly to configure a complex business process by service components. Other researchers also take the ontology technology to help information system design. Ana Simonet [Simonet, 11] presents an ontology-based data model called ISIS which contains three meta-concepts: *concept, binary relation* and *ISA* relation. By using ISIS, well formatted domain ontology and a set of use cases are provided for the design of information system. Although the proposed domain ontology contains behavioural properties, it is still difficult to describe the operational semantics for a web service. Petr Kremen [Kremen and Kouba, 12] proposes a methodology for designing ontology-based software applications that make the ontology possible to evolve while being exploited by one or more applications at the same time. Chow et al. [Chow et al., 09] present an ontology-based information sharing application using Service-Oriented Database System (SODB) to facilitate data integration. By taking the advantage of mathematical equivalence relations in domain ontology, the application could map the data sources into appropriate ontology and facilitate dynamic query composition across data sources.

From the above literatures, we could find that researchers have made lots of improvements for semantic modelling on entities and operations for ontology-based service composition. But the existing methods of operational semantic modelling are mainly focused on describing operational semantics by IOPE rather than operation itself. Moreover, a semantic model which could combine structural and operational semantics is still left to be proposed.

3 Semantic Service Model

In this section, we introduce a formal definition of SRSM (Semantic Resource Service Model) in order to model the entities, relations, operations and data storage for ontology-based semantic service composition. And depends on SRSM, an automated service generating approach is proposed to transfer the resource into RESTful web service.

3.1 Overview of our approach

Semantic Resource Service Model is used to formally describe the concepts, entities, relationships, functions and persistence of the resource service. For service semantics, SRSM uses OWL [Antoniou and van Harmelen, 09] ontologies to annotate the features of service. For service relationships, SRSM defines a formal structure type which called Entity-Oriented Resource (EOR) to compose small resources into a composited one to meet requirements. For service operations, SRSM defines another structure type which called Transition-Oriented Resource (TOR) to implement complex functions in resource service. For resource persistence, SRSM configures the resource boundary information to clearly distinguish the borders among the different resources in physical data level. For service composition, SRSM combines structural semantics and operational semantics of a service to compose sequence services by semantic reasoning. Figure 1 shows the overview of SRSM.

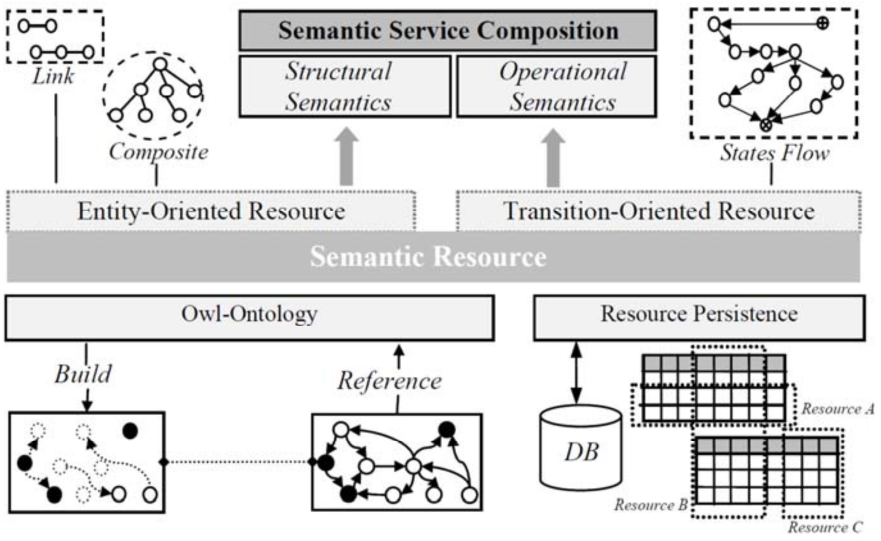


Figure 1: Semantic Resource Service Model - SRSM

As Figure 1 shows, the core part of SRSM is TOR and EOR (EOR supports structural semantic reasoning as well as TOR supports operational semantic reasoning) which is based on Semantic Resource. Thus, Semantic Resource should include full meta-data about TOR and EOR. In our approach, Semantic Resource meta-data is represented in XML format. Entity-oriented and transition-oriented resources have same representation of attributes, but different in the representation of functions. The table 1 shows the main XML tags of meta-data with its means.

3.2 Entity-oriented Resource

A useful approach to organizing knowledge is to consider each “entity” a resource and then to find a simple way to represent and share the resource [Waters et al., 09]. In the traditional process of business modelling, entities are identified to represent the object which manipulated by a task or process. An entity only contains the information and structure of real object in a business environment. For example, ‘Order’, ‘Customer’ and ‘Items’ obviously are the entities in an online shopping system, in opposite, ‘SubmitOrders’, ‘MergeOrders’ are not entities. For the implementation, an entity should support the process of obtaining, operating and saving so as to meet the business requirement. Entity-oriented resource is designed to formally encapsulate the entity from model level into implement level with the capability of obtaining, operating and saving.

Semantic Resource			
Description	Detail description about the resource	Data Base Config	
StateSpace	A reference that assign the resource's state transfer paths.	DBURL	Standard Database URL.
ResourceURI	Assign the Resource's Identifier	User	Database user name for remote connecting
ResName	Assign the Resource's Name, and it will appear in URI	Password	Database user PSD for remote connecting
IsBase	Indicate that the resource is single or composite resource	Driver	A full java package refer from standard JDBC driver
ResourceSuper	Configure the OWL super class for this resource		
Join Relationship		Attributes	
InverseJoin	The value 'true' indicates that the resource will be joined in N:1 style, else in 1:N style	ColName	Assign the name. Usually, it use lower case for the compatibility of different database columns
SuperProperty	Configure the OWL super ObjectProperty for this relation	ColSuper	Configure the OWL super DataProperty for this column
JoinRelation	Describe the relations in literal.	ColType	Column types refer from database columnType
JoinURI	The target resource URI	ColAlias	Attribute name for display
Transition-Oriented Resource		Entity-Oriented Resource	
Function	Indicate the entry-point and end-point of a transition	Condition	The condition is used to restrain the instance of the resource
Condition And Expression	Validate the resource instance whether the states of the instance meet the conditions	Symbol	Truth condition between l-value and r-value, e.g. '>', '=', and also include 'and', 'or','!' etc.
Behaviour And Expression	The basic operations of resource such as GET, PUT are defined in XML format. Other behaviours are described in script-like language executed in the explain engine.		

Table 1: Semantic Resource in XML representation

3.2.1 Single Resource

Most resources like ‘customer’, ‘product’, ‘item’ and ‘staff’ are easily identified from a business scenario not only in logic level but also in physical storage level. They are independent resources and usually treated as a kind of master data [Loshin, 08]. The Figure 2 gives a typical single resource in our framework.

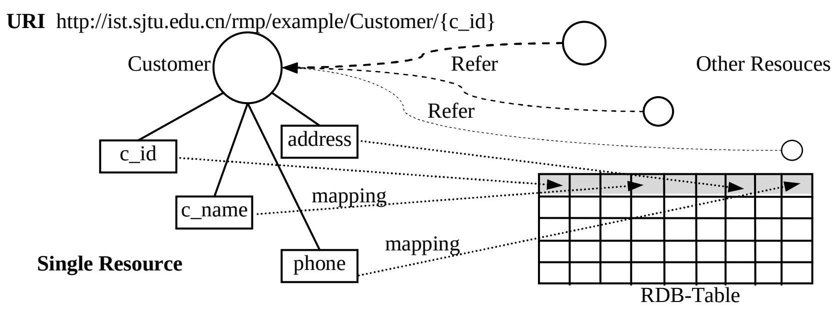


Figure 2: the single resource

Every single resource can be address by a unique identifier called URI. The prefix part of URI is consisted of http domain path, and the local part is mapped primary key from RDB tables. Additional, one single resource cannot map to one or more tables that means it just a part of single table.

3.2.2 Composited Resource

Although most of resources are modelled as single resource, some of the resources are more complex and difficult to organize as a single resource. Consider the entity of ‘Order’, it is consisted of attributes, the items belong to *Order* and the customer who has the *Order*. Indeed, a composited resource is consisted of many other entities or resources which contain “one to one” or “one to many” relationships. The Figure 3 shows an example of composited resource.

The element ‘key-transfer’ in Figure 3 is a mapping function which is used to figure out the join target key from a key mapping set. The key map contains two keys: domainKey and rangeKey, it is used to match the list of rangeKey which could specify the joined resource by specified domainKey. Besides, every resource has its own DB information that means the rangeKey can be redirected to another data source in a distributed environment.

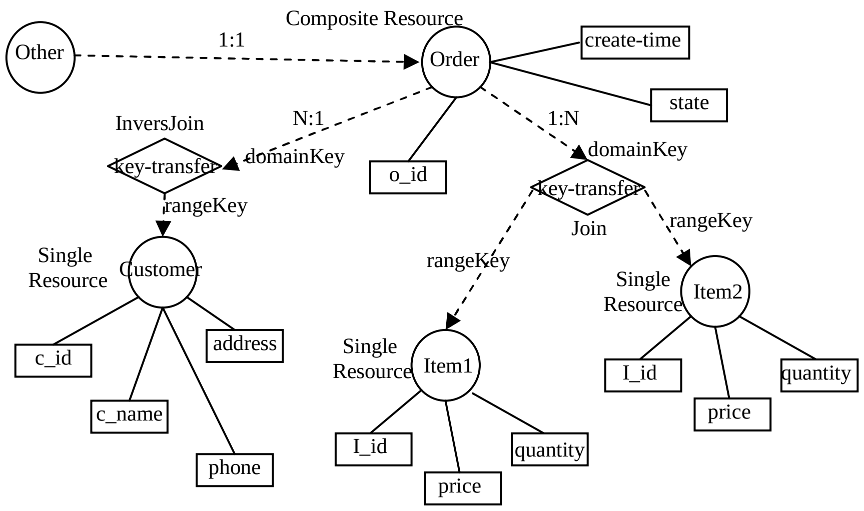


Figure 3 the composite resource

3.3 Transition-oriented Resource

Not all the resources can be represented as entities in the process of information system developing. Some of the resources have special features about function or operation, such as ‘ContactsBackup’, ‘ContactsMerge’ and ‘AccoutTransition’. Although these resources are addressed by a unique URI and persistent in distributed DB environment same as Entity-Oriented Resource, it is aimed to do a task or action in a quite different way. It starts an action when user POST(create) this kind of resource and stop the action when user DELETE(cancel) the resource. In a word, these resources are defined as Transition-Oriented Resource that consumes and manipulates other Entity-Oriented Resource.

3.3.1 Resource state representation

The features of the resources are mainly appeared in the attributes of the resources. Each attribute has one value domain that regulates the scope of available values for the attribute. The different values of attributes represent different states of the resources. However, depends on business requirement, a resource’s state only appeared in some attributes in which have a finite value domain, and these value domains can be modelled as a state matrix with the transition functions. In contrary, there is no sequence for state change if the attribute value domain is infinite.

As Table 2 shows, resource’s state space is defined as the set ASS= $\{s_i \mid i=0, 1\ldots, n\}$. n=|ASS|>0. The $n \times n$ Transfer Matrix M_R for the resource can be derived according to the finite value domain of resource attribute. If there is state transfer $s_i \rightarrow s_j$, then the element in Matrix is set as $a_{ij}=1$, otherwise $a_{ij}=0$.

	s_0	s_1	s_2	s_3	s_4	s_n
s_0	1	1	0	0	0	0
s_1	0	1	1	0	0	1
s_2	0	0	1	1	0	0
s_3	0	0	0	1	1	0
s_4	0	0	0	0	1	1
s_n	0	0	0	0	0	1

Table 2: resource state represented in Transfer Matrix

Furthermore, Define the state of resource R in situation S_i is represented as a Situation Vector SV, $SV_{(R,\ Si)} = M_R[*,i]$. Given the $SV_{(R,\ i)}$ and the State Transfer Matrix M_R , the acceptable Mask Vector $MV_{(mask,\ i)}$ can be deduced as $V_{(mask,\ i\ )} = V_{(R,\ i)} \times M_R$. For each $V_{(R,\ i+k)}$, if $V_{(R,\ i+1)} \times MV_{(mask,\ i\ )}^T > 0$, the state $V_{(R,\ i+1)}$ is acceptable, else the state $V_{(R,\ i+k)}$ is unacceptable. The results set can be converted to a transfer diagram show in Figure 4.

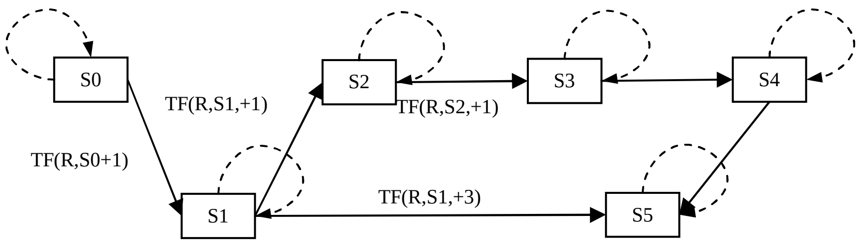


Figure 4: State transfer graphs

According to the State Transfer Matrix M_R, the general transfer function $TF(R_i, s_j, k)$ is used to change the resources state in k step.

4 Semantic Service Composition

In this section, a novel service composition method is presented by combining structural semantics and operational semantics of the service. Section 4.1 introduces the structural semantics and Section 4.2 presents the operational semantics of a service. Then, Section 4.3 combines structural semantics and operational semantics to implement service composition.

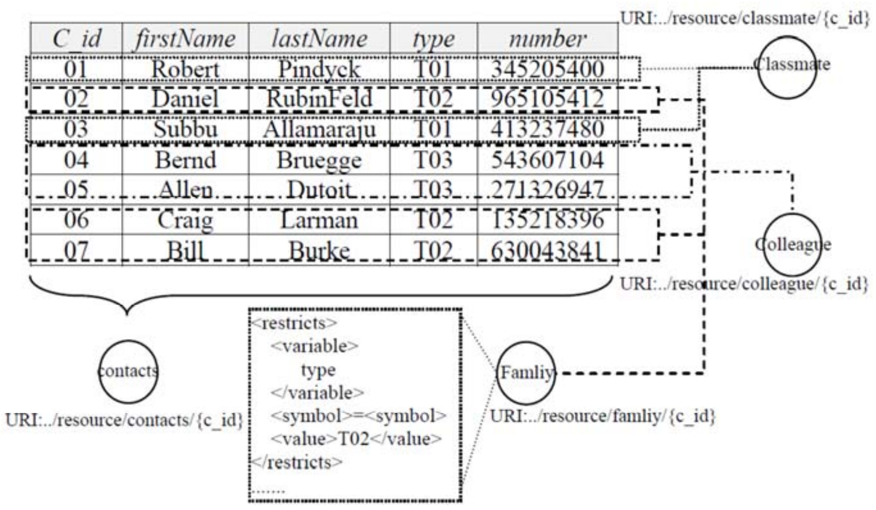
4.1 Structural Semantics

Structural semantics are the key elements that determine the input/output matching between services. Heterogeneous input/output of a service can be matched if the “equivalent”, “include” or “sub-set” relations can be found between input and output.

In this section, two structural semantics are introduced that are resource specialization and resource generalization within a scenario of phone book service.

4.4.1 Resource Specialization

In a phone book service, the main resource is 'Contacts' which contains the whole person information. However, it seems that everyone prefer group the contacts in family, colleague or classmate than just search them in a single level. Classify one resource into several more specific resources is a kind of implementation of resource specialization. In our approach, the resource meta-model defines the resource boundaries by configuring the resource restrictions to distinguish different resource in data layer. The figure 7 shows the mechanism of TYPE I specialization.



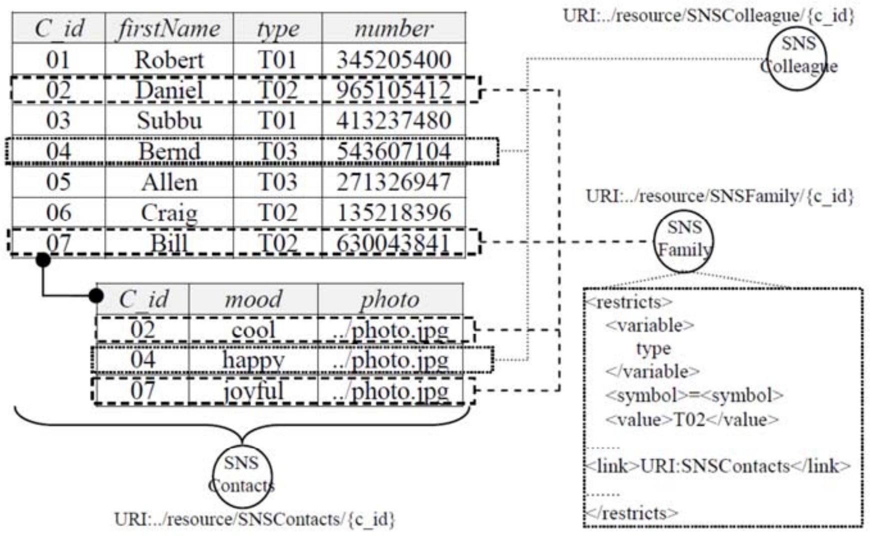
C_id	*firstName*	*lastName*	*type*	*number*
01	Robert	Pindyck	T01	345205400
02	Daniel	RubinFeld	T02	965105412
03	Subbu	Allamaraju	T01	413237480
04	Bernd	Bruegge	T03	543607104
05	Allen	Dutoit	T03	271326947
06	Craig	Larman	T02	135218396
07	Bill	Burke	T02	630043841

```
<restricts>
  <variable>
     type
  </variable>
  <symbol>=<symbol>
  <value>T02</value>
</restricts>
......
```

URI:../resource/classmate/{c_id}
Classmate
Colleague
URI:../resource/colleague/{c_id}
contacts
URI:../resource/contacts/{c_id}
Famliy
URI:../resource/famliy/{c_id}

Figure 7: the Type I of Resource Specialization

The TYPE I specialization can classify the instance of resource according different attributes values. However, it cannot identify the resource which has more attributes. Thus, the TYPE II specialization is used to link the extra attributes to define a specialized resource that shows in Figure 8.



C_id	firstName	type	number
01	Robert	T01	345205400
02	Daniel	T02	965105412
03	Subbu	T01	413237480
04	Bernd	T03	543607104
05	Allen	T03	271326947
06	Craig	T02	135218396
07	Bill	T02	630043841

C_id	mood	photo
02	cool	../photo.jpg
04	happy	../photo.jpg
07	joyful	../photo.jpg

```
<restricts>
   <variable>
      type
   </variable>
   <symbol>=<symbol>
   <value>T02</value>
......
<link>URI:SNSContacts</link>
......
</restricts>
```

URI:../resource/SNSColleague/{c_id}
SNS Colleague
URI:../resource/SNSFamily/{c_id}
SNS Family
SNS Contacts
URI:../resource/SNSContacts/{c_id}

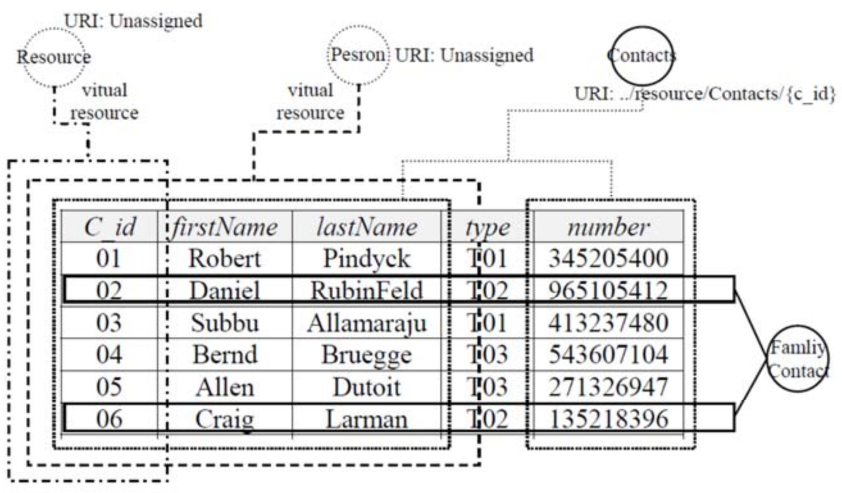
Figure 8: the Type II of Resource Specialization

After specialization, different resource, which can be used as input/output of a service, has been organized together by “sub-set” relationship. For example, as figure7 shows, FamilyContacts and ColleagueContacts are sub-set of Contacts that means FamilyContacts and ColleagueContacts are matching the input of Contacts.

4.4.2 Resource Generalization

Consider a service that allows large scale input/output types which include multiple types (e.g. a LocationSearch service allows multiple input types that can be “city”, “country” and “region”). The best way to implement this is to find a super type which includes all the allowed types.

Generalized Resource is a resource which holds less restriction than specialized one - i.e. Generalized Resource is an abstract resource. In our method, the attributes of resource are treated as the restriction to define Generalized Resource: giving two resources A, B and the restriction R_A, R_B, A is Generalized Resource of B if and only if $R_A \in R_B$. The figure 9 shows the Generalized Resource in phone book service.



C_id	*firstName*	*lastName*	*type*	*number*
01	Robert	Pindyck	T01	345205400
02	Daniel	RubinFeld	T02	965105412
03	Subbu	Allamaraju	T01	413237480
04	Bernd	Bruegge	T03	543607104
05	Allen	Dutoit	T03	271326947
06	Craig	Larman	T02	135218396

Figure 9: the fundamental of resource generalization

It is necessary to note that the generalized resources are not unique, and even some of them are not really existed in current data environment. Instead, these inexistent generalized resources are described as a Virtual Resource. In Section 4.2, we will discuss Virtual Resource in more details.

4.2 Operational Semantics

Although we have identified different kind of resources from shared or distributed data regions such as the resource 'Contacts', 'FamilyContacts', 'SNSFamily' and 'Persons' in Section 4.1, it seems there have some potential relations between these resources. Fortunately, one can easily figure out that 'FamilyContacts' and 'SNSFamily' is a kind of 'Contacts' by further analysis. But, it is hardly to find out that 'SNSFamily' is also a kind of 'FamilyContacts' and 'Persons' in the current data environment.

In this section, we will introduce a formal method to describing and reasoning the two types of resources by using OWL ontology.

4.2.1 Resource Relation Reasoning

There is no direct way driver from one place to another just because of the natural barriers such as mountains, rivers and oceans. The principle of resource relation reasoning is describing the resource itself and its virtual resource as elaborate as possible rather than directly declaring the relations between the resources.

Definition 1. Virtual Resource. R_i is the restriction of Resource$_i$. And VR_j is a restriction combination. VR_j is a Virtual Resource (VR) if and only if $VR_j \in \exists R_i$.

VR is a resource but usually used to describe an aspect of real resource. One resource always contains many aspects such as resource Teacher contains some aspects about Person, Educator and Presenter. Each aspect is treated as a VR with no name assigned until there is real resource equivalent to this VR. Based on VR, the resource relation reasoning can be addressed as follow.

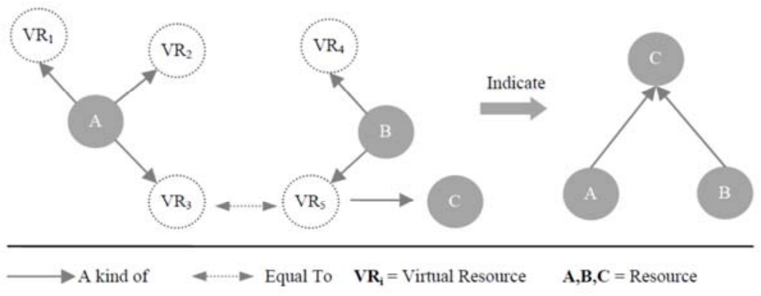
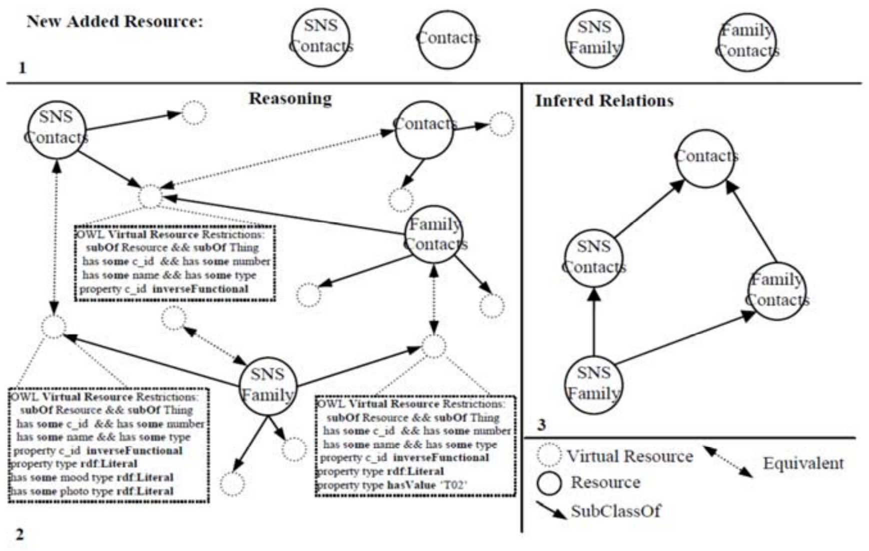


Figure 10: Virtual Resources in Resource Relation Reasoning

As the Figure 10 shows, resource A has three aspects that described by VR_1, VR_2 and VR_3. Resource B has two aspects that described by VR_4 and VR_5.Resource A and B is a kind of C because VR_3 equivalent to VR_5, and VR_5 is a sub-set of resource C. Back to the phone book service, the Figure 11 gives a more complex resource relation reasoning.



```
OWL Virtual Resource Restrictions:
  subOf Resource && subOf Thing
 has some c_id  && has some number
 has some name && has some type
property c_id  inverseFunctional
```

```
OWL Virtual Resource Restrictions:
  subOf Resource && subOf Thing
 has some c_id  && has some number
 has some name && has some type
property c_id  inverseFunctional
property type rdf:Literal
has some mood type rdf:Literal
has some photo type rdf:Literal
```

```
OWL Virtual Resource Restrictions:
  subOf Resource && subOf Thing
 has some c_id  && has some number
 has some name && has some type
property c_id  inverseFunctional
property type rdf:Literal
property type hasValue 'T02'
```

Figure 11: Example of Resource Relations Inference

Entity-Oriented Resource is treated as input/output of Transition-Oriented Resource. Based on relation reasoning, heterogeneous Entity-Oriented Resource can be matched even the structure of the resources are quite different.

4.2.2 Resource Process Reasoning

Although most operations are dynamic things that are hardly represented in ontology or RESTful service, it can be transformed to static things by describe the operation as a Transition-Oriented Resource. For example, the operation 'pay the orders' is a dynamic thing, but it can be transformed into a static resource 'OrderPayment' [Zhao and Doshi, 09]. In our method, such resources are considered as transition-oriented resource. In order to automatically compose sub-processes to realize a complete transaction, transition-oriented resource need to support process reasoning. It not only requires IO (input and output) reasoning, but also needs PE (pre-conditions and effects) reasoning. Furthermore, IO reasoning is used to decide whether an instance matches the input/output (e.g. Type A fits the input requirement because A is an equivalent-type or sub-type of input required [see Section 4.1]). In contrary, PE reasoning is used to decide what conditions should be satisfied before process start and what states should be changed when process execute.

Consider the Transition-Oriented Resource 'OrderPayment' which is used to pay the order. It can be divided into three sub-processes that are CheckInventory, CheckBalance and PayOrder. Each sub-process has its own IOPE conditions that are described in OWL-SWRL format and shown in Table 2.

Transition-Oriented Resource "/OrderPayment/{id}"		
Input/**O**utput Entity-oriented	**P**re-condition OWL-SWRL	**E**ffects OWL-SWRL
(1) Check Inventory		

Order/Order	hasType(?order, #Input)
	hasType(?order, #Output)
	hasType(?order, #Order)
		referInventory(?order ?inventory)
		hasQuantity(?order, ?quantity)
		lessThan(?quantity, ?inventory)
		indicate →
		stockState(?order,'checked')

(2) Check Balance		

Acc/Acc	hasType(?order, #Order)
	bindAcc(?order,?acc)
	hasType(?acc, #Acc)
	hasType(?acc, #Input)
	stockState(?order,'checked')
		totalPrice(?order, ?total)
		balance(?acc, ?balance)
		lessThan(?total, ?balance)
		indicate →
		balanceState(?order,'checked')

(3) Pay Order		

Order/Order	hasType(?order, #Input)
	hasType(?order, #Output)
	hasType(?order, #Order)
	balanceState(?order,'checked')
		hasType(?order, #Order)
		totalPrice(?order, ?total)
		bindAcc(?order, ?acc)
		hasType(?acc, #Acc)
		balance(?acc, ?balance)
		indicate →
		balance(?acc, ?balance-?total)
		orderState(?order, 'paid')

Table 2: Semantics of IOPE in Transition-Oriented Resource

As the Table 2 shows, the final state of "OrderPayment" is *balance (?acc,?balance-?total)* and *orderState(?order, 'paid')* which is the result of PayOrder. Furthermore, before to start PayOrder, the state *balance(?order, 'checked')* which is the result of CheckBalance must be satisfied first. Then, before CheckBalance start, the state *stock(?order, 'check')* which can be obtained from CheckInventory must be satisfied. Thus, according IOPE descriptions in transition-oriented resource, the process flow of "OrderPayment" can be inferred as CheckInventory-> CheckBalance-> PayOrder.

4.3 Service Composition

It is worth to note again that the core idea of our approach is combining structural and operational semantics for service composition. Structural semantics are used to match the input and output of service (i.e. input/output reasoning) as operational semantics are used to express the functions of service (i.e. pre-condition and effects reasoning).

Definition 2. Start State. Give the input requirements R_{in} and pre-condition C_P of a service. The Start State (SS) is a restriction set $R_{in} \cup C_P$.

Definition 3. End State. Give the output requirements R_{out} and effects C_e of a service. The End State (ES) is a restriction set $R_{out} \cup C_e$.

According to Definition 2-3, a service is considered as state transition function $\mathcal{F}_s(state)$ that transfers the state of the service: $ES \leftarrow \mathcal{F}_s.(SS)$. Thus, based on proposed method, service composition can be described as: giving the entry-point (SS) and the end-point (ES) of the requirements, finding out a state transition sequence ($\{\mathcal{F}_{s1}\,\mathcal{F}_{s2}\,\mathcal{F}_{s3}\,\ldots\}$) that transfers the state from SS to ES. This can be classified as a regression problem that addressed in following algorithm.

```
Input: SS(in,P), ES(out,E) and Service set S={$s_1$, $s_2$, $s_3$..}
$S_{final}$ ← find all final services in S where $ES_{final}$ and
        ES(out,E) matched.
For all services $s_{fi}$ in $S_{final}$ do
$PC_i$ ← Regressing the service $s_{fi}$ while $SS_{next}$ and SS(in,P)
      matched
End for
Output: PC,i.e. Possible Composition path set.
```

The output of above algorithm is a path set of possible composition. The matching on ES/SS(in/out) is using Structural Semantics reasoning [see Section 4.1] and the matching on ES/SS(P/E) is using Operational Semantics reasoning [see Section 4.2].

5 A Case Study

Problem Description. In this case study, we aim to illustrate the following business process scenario: Prior to donation agreement taking effect, both donors and recipients should conduct a series of approval processes. Firstly, donors should propose an agreement with detailed restraints and send to recipients. Secondly, recipients review the agreement and decide whether to accept the agreement by the president of foundation. Then, both donors and recipients sign the agreement. At last, the states of agreement will be changed from "proposed", "accepted", and "unsigned" to "effect". The following steps show how we implement this business process by

SRSM.

Step1. Configuring business object as Entity-Oriented Resource.

In this case, we have recognized three main business entities that include agreement, donor and recipient. At the beginning, each entity is described by service meta-model and assigned a unique URI such as "**/foundation/donor/ {id}*". Then, the relationships among these entities are created (e.g. composing donor and recipient as a part of Agreement) [see Section 3.2 and Section 4.1]. The following figure shows the service meta-model of these business entities.

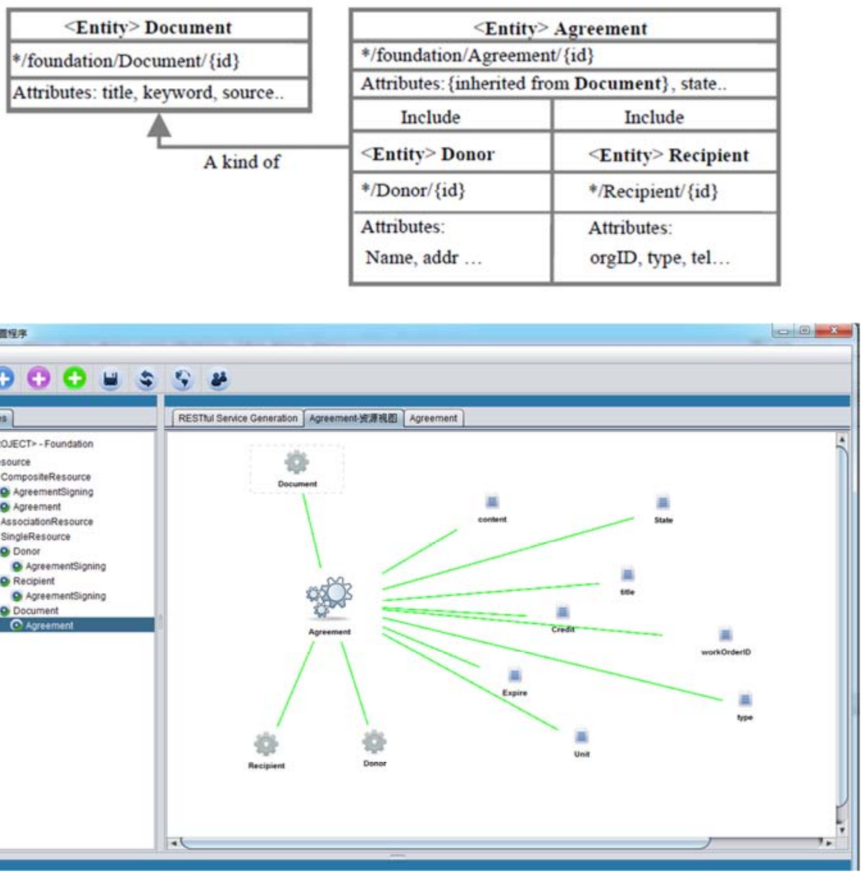


Figure 12: Service meta-model of these business entities

After service modeling, the business entities actually becomes a resource service which means user can access these entities in RESTful style. Moreover, the resource "Agreement" is a composited service which includes "Donor" and "Recipient". It is very useful when user want to find out the agreement's donors and recipients in a business process by using only once GET method on "Agreement".

Step2. Configuring business process as Transition-Oriented Resource.

Agreement signing is a typical business process in school foundation. It can be divided into a series of small business processes such as agreement proposing, review, signing, etc. According to transition-oriented resource [see Section 3.3 and Section 4.2], we can define a resource "AgreementSigning" as a transaction service so as to execute the processes of agreement signing. The following table shows the definition of "AgreementSigning".

<Transition> AgreementSigning			
URI	*/foundation/AgreementSigning/{id}	**Attributes**	startTime, name, InputResource, **propose, review, sign**…
Transition Function	**Semantics**		
	Precondition	**Effects**	
Propose	<owl-swrl> hasType (?x, Invoker) hasType(?x, Donor) </owl-swrl>	● POST */foundation/Agreement/{id} ● Semantic change (automatic): <owl-swrl> hasType(?x,Agreement), hasState(?x,?y), hasValue(?y,null) ➔ hasValue(?y,'proposed') </owl-swrl>	
Review	<owl-swrl> hasType (?x, Invoker) hasType (?x, recipient) hasType (?x, president) hasType (?y, Agreement) hasState (?y, ?z) hasValue(?z, 'proposed') </owl-swrl>	● PUT */Agreement/{id}?state=accepted ● Semantic change (automatic): <owl-swrl> hasType(?x,Agreement), hasState(?x,?y), hasValue(?y,'accepted') ➔ hasValue(?y,'unsigned') </owl-swrl>	
Sign	<owl-swrl> hasType (?x, Invoker) hasType (?x, Donor || Recipient) hasType(?y, Agreement) hasState(?y, ?z) hasValue(?z, 'accepted') </owl-swrl>	● PUT Agreement(signature) ● Semantic change (automatic): <owl-swrl> hasType(?x,Agreement), hasState(?x,?y), hasValue(?y,'accepted') hasSignature(?x, Donor) hasSignature(?x, Recipient) ➔ hasValue(?y,'effect') </owl-swrl>	

Table 3: Transition-Oriented Resource for AgreementSigning

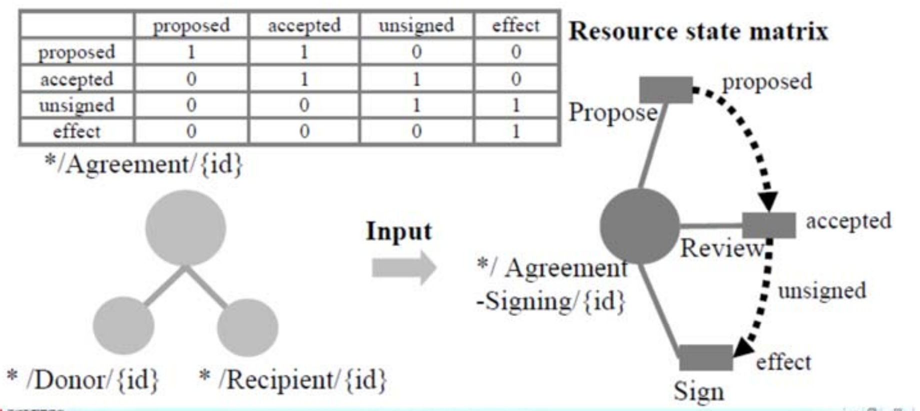
Indeed, "AgreementSigning", defined by SRSM, is a composited RESTful service in which contains three sub-processes: proposing, reviewing and signing. Each sub-process has its own preconditions and effects which described in OWL-SWRL format. Before each process start, ontology reasoner will check the precondition. And after process finished, ontology reasoner will change the resource's states according to effects description. Moreover, in order to control the signing process, the four methods (POST, PUT, DELETE and GET) are allowed in

“AgreementSigning” (not all transition-oriented resources provide four methods). The means of each method are explained as follows:

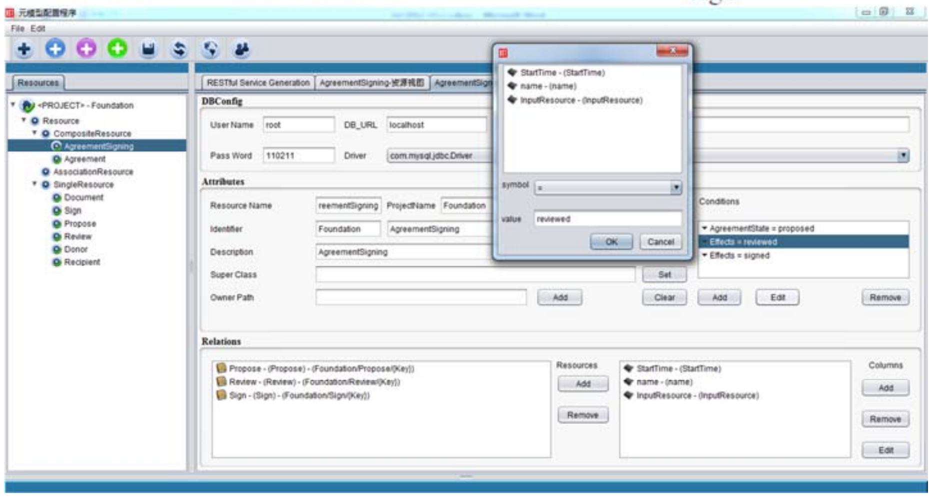
- POST: system checks whether the invoker is a “Donor”, then starts a new transaction – i.e. *POST */AgreementSigning/{new_id}*.
- GET: system returns an existing “AgreementSigning” to user.
- PUT: user can change the attributes of “AgreementSigning” to invoke a specified process if all preconditions are satisfied (e.g. *PUT */AgreementSigning/{id}, attributes change: review= true*).
- DELETE: system will cancel the transaction.

Step3. Composing services by semantic reasoning .

According to resource state matrix [see Section 3.3], structural semantics [see Section 4.1], operational semantics [see Section 4.2] and service composition [see Section 4.3], the state transition sequence can be addressed. The figure 13 shows the transition flow of agreement signing.



	proposed	accepted	unsigned	effect
proposed	1	1	0	0
accepted	0	1	1	0
unsigned	0	0	1	1
effect	0	0	0	1



<PROJECT> - Foundation
Resource
CompositeResource
AgreementSigning
Agreement
AssociationResource
SingleResource
Document
Sign
Propose
Review
Donor
Recipient

DBConfig
User Name: root
DB_URL: localhost
Pass Word: 110211
Driver: com.mysql.jdbc.Driver

Attributes
Resource Name: reementSigning
ProjectName: Foundation
Identifier: Foundation AgreementSigning
Description: AgreementSigning
Super Class:
Owner Path:

StartTime - (StartTime)
name - (name)
InputResource - (InputResource)
symbol: =
value: reviewed

Conditions
AgreementState = proposed
Effects = reviewed
Effects = signed

Relations
Propose - (Propose) - (Foundation/Propose/{Key})
Review - (Review) - (Foundation/Review/{Key})
Sign - (Sign) - (Foundation/Sign/{Key})

Resources

StartTime - (StartTime)
name - (name)
InputResource - (InputResource)

Columns

Figure 13: resource state transition flow of agreement signing

As figure 13 shows, a donor wants to donate money to school foundation. He must follow a donation process to make a donation deal. Firstly, he will POST "AgreementSigning" to start the donation transaction. Then, he will PUT the "AgreementSigning" to invoke the proposing process (*PUT */AgreementSigning/ {id} change: propose=true*). When an agreement has proposed, the president of foundation will review the agreement. He will GET "AgreementSigning" to read the agreement contents and then invoke the review process (*PUT */AgreementSigning/{id}?change: review=true*).When the agreement has accepted, both donor and recipient will invoke signature process to accomplish a donation deal (donor and recipient, *PUT */AgreementSigning/ {id} change: sign=true*).

Step4. Generating resource services

Until by now, Entity-Oriented resources (i.e. *Document, Agreement, Donor, and Recipient*), Transition-Oriented resources (i.e. *AgreementSigning*) and resource compositions (i.e. *Agreement* signing process) are ready. The rest of the task is generating RESTful web service from resource..

According to service model [see Section 3], Entity-Oriented Resources are transformed into Resource service by its URI. For example, resource *Donor* is transformed into service "*{domain}/{Unit}/Donor/{id}*". In contrary, one Transition-Oriented resource is transformed into series sub-services. For example, resource *AgreementSigning* is transformed into "*{domain}/{Unit}/Propose/{id}*", "*{domain} /{Unit}/Review/{id}*" and "*{domain}/{Unit}/Sign/{id}*". The following figure shows system snapshot of services generating from Entity-Oriented Resource and Transition-Oriented Resource.

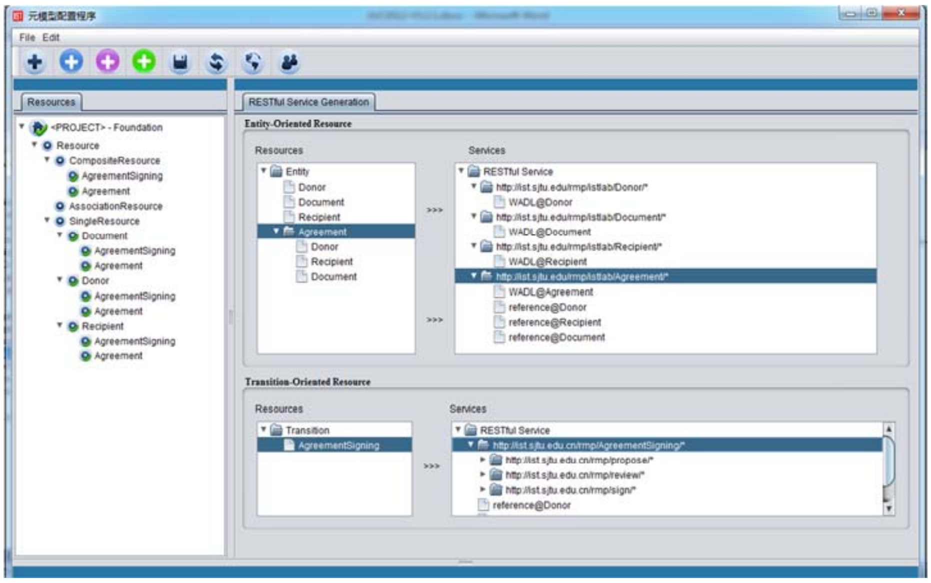


Figure 14: System Snapshot of Service Generating

6 Comparison and Discussion

6.1 Comparison

NetKernel is a resource-oriented information services and integration platform towards application configuration [Geudens, 12]. From addressing elements (resources, services and code) with a URI to loosely couple the internals of applications, NetKernel can make applications more flexible and powerful. By sharing interoperable data models (resource models), NetKernel find a way let the process of business solution building more easily. The details of comparison are shown in Table 2.

Feature	SRSM	NetKernel
Model Element	Entity-oriented Resources Transition-oriented Resources	Identifiable Resources Computational Modules Accessors
Resource Configuration	Semantic Resource Meta-Model based resource configuration with XML tags	Interoperable Data-model based resource configuration with code fragment
Entity Definition and Access Method	Single Resource Composite Resource REST Style Service Access	UNIX Style Resource Composition RESTful & SOAP Service Access
Operation Definition	Transition-oriented Resource Configurable XML tags and Scripts	Computational Modules Operational code fragment Dynamic programs are also modelled as resources
Operation Call	RESTful Service with Transition-oriented Resource	RPC with SOAP Service RPC with Dynamic programs
Structural Semantics	Entity relations and features are defined in meta-model, and described in owl format	Not mentioned
Operational Semantics	The processes of an operation are represented by owl-ontology restrictions such as Property Chains, InverseProperty, FunctionalProperty etc.	Not mentioned
Service Configuration	Referred to MVC pattern, RESTful Services are mapped to entity or transition resource configuration	Based on Resource Oriented Computing (ROC), resources, services and code are configured.

Table 3: Similar system comparison

As Table 3 shows, comparing features of SRSM with those of NetKernel from three aspects: entity structure, operation, and configuration, SRSM provides a combined structural and operational semantics model to reduce the complexity of information system constructing, and utilizing owl-ontology to carry the semantics of resource model to make the ability of resource retrieving and manipulating more powerful.

6.2 Discussion

The case study and comparison has shown that SRSM contains many enhanced features and functionalities. However, in today's business environment, ever-changed requirements and varied business expansions force enterprise to consider the usability, extensibility and adaptability as much as functionality of a service. Thus, in this section, we will discuss our approach in these three aspects.

1) Usability

Firstly, it is accessible for both service developers and service consumers who know the business well since all business entities are encapsulated as resource services in SRSM. Moreover, User can easily compose different resources into a composited resource to meet the business requirements without concerning any implement details (e.g. *Order* is a composited resource which composes of *Item*, *Owner and Shipment*).

Secondly, beyond business entities, transactions in which contain actions or operations are often appeared in business scenarios. Based on Transition-Oriented Resource [section 3.3], users are allowed to only concern the preconditions and effects (post-conditions) of a transaction. The business processes can be automatically generated by ontology reasoner after preconditions and effects inferring.

2) Extensibility

User wants to build more entities or functions without any changes in original system when their business has extended. In SRSM, every entities and functions are considered as resources which can be realized as pluggable components. Once a resource has defined, its meta-model becomes a concept described by ontology. The new added entities are directly plugged into original system if the entities are totally new. Otherwise, by means of ontology reasoning, the new added entities are mapping to specialized or generalized resources of existing resources (e.g. new added entity "*Contract*", with its own characteristics, becomes a specialized resource of existing resource "*Document*").

3) Adaptability

Unlike extensible system, adaptable system requires composited service can do corresponding reactions when business requirement has changed.

Towards business entities changing, if an existing resource has been deleted, SRSM can automatically find a specialized resource to instead of deleted one by using conceptual semantics inference. Moreover, if the attributes of an existing resource has been changed following new business requirement, it will be re-inferred and may become a specialized or generalized resource of others.

Towards business processes changing, if there some new processes have been added into an existing transaction, then SRSM will re-generate the transaction flow by inferring the preconditions and effects (post-conditions) of each process.

7 Conclusion

Based on SRSM, we proposed a comprehensive solution for semantic entity composition, semantic operation modelling and resource-oriented service for resource-oriented services composition. Considering various requirements in services composition, SRSM not only provides an ontology-based structural semantic for entity modelling, but also provides an ontology-based operational semantic and RESTful service for process execution. Moreover, both structural semantics and operational semantics are modelled in resource meta-model which can be conveniently accessed by composited RESTful web service. Therefore, structural semantics and operational semantics are seamlessly combined so as to meet the variable requirements of resource-oriented services composition.

Our further tasks will focus on compositing the transition-oriented resource in semantic level with the aim to configure complex operation by resource composition. Take the advantage of operational semantics in transition-oriented resource, the resource composition will not only consider the IOPE features, but also analyse the detail semantics of operation itself.

Acknowledgements

This research is supported by the National Natural Science Foundation of China under No. 71171132, and the National High Technology Research and Development Program of China ("863" Program) under No.2008AA04Z126.

References

[Andrea and Egenhofer, 03] Rodriguez, M.A., Egenhofer, M.J.: Determining Semantic Similarity among Entity Classes from Different Ontologies, IEEE transactions on knowledge and data engineering, 2003, Volumes 15, pp 442-456

[Antoniou and van Harmelen, 09] Antoniou, G., van Harmelen, F.: Web Ontology Language: OWL, International Handbooks on Information Systems, pp 91-110, 2009 [Bradford, 10] Bradford, R. B.: Entity refinement using latent semantic indexing, International Conference on Intelligence and Security Informatics (ISI), 2010, pp 126-128

[Cai et al., 12] Cai, H., Xu, B., Bu, F.: A Conceptual Ontology-based Resource Meta-Model towards Business-driven Information System Implementation, Journal of Universal Computer Science, vol. 18, no. 17, pp 2493-2513, 2012

[Chow et al., 09] Chow, T., Tsai, W.-T., Balasooriya, J., Bai, X.: Ontology-based Information Sharing in Service-Oriented Database Systems, International Conference on Services Computing, 2009, pp 276-283

[Feiyu et al., 12] Lin, F., Sandkuhl, K., Xu, S.: Context-based Ontology Matching: Concept and Application Cases, Journal of Universal Computer Science, vol. 18, no. 9, pp 1093-1111, 2012

[Geudens, 12] Geudens. T.: (2012, May 11), Resource-Oriented Computing with NetKernel: Taking REST Ideas to the Next Level. Available: http://shop.oreilly.com/product /0636920024460.do

[Hadley, 06] Hadley, J.: Web Application Description Language (WADL), 2006, Available: http://wadl.dev.java.net/

[Hai, et al., 11] Hai, R., Theißen, M., Marquardt, W.: An ontology based approach for operational process modelling, Advanced Engineering Informatics, 2011, Volumes 25, pp 748-759

[Hitzler et al., 09] Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P.F., Rudolph, S.: OWL 2 Web Ontology Language Primer, 2009, Available: http://www.w3. org/TR/ 2009/WD-owl2-primer-20090421/#Property_Chains

[Horrocks et al., 04] Horrocks, I., Patel-Schneider, P. F., Boley, H., Tabet, S., Grosof, B., Dean, M.: Swrl: A semantic web rule language combining owl and ruleml, http://www.w3.org/Submission /SWRL/, 2004.

[Kremen and Kouba, 12] Kremen, P., Kouba, Z. (2012). Ontology-Driven Information System Design.Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on, 42(3), 334-344.

[Liu, 11] Xumin Liu, X., Liu, H.: Constructing Operation-Level Ontologies for Web Services, IEEE International Conference on Web Services (ICWS), 2011, pp 716-717

[Loshin, 08] Loshin, D.: Master Data Management, Morgan Kaufmann Publishers Inc, ISBN: 0123742250, 2008

[Pérez et al. 10] Pérez, S., Durao, F., Meliá, S., Dolog, P., Díaz, O.: RESTful, Resource-Oriented Architectures: A Model-Driven Approach, Web Information Systems Engineering, 2010, Volume 6724, pp 282-294

[Plotkin, 04] Plotkin, G.D.: A Structural Approach to Operational Semantics, Program, 2004, Volumes 60-61, pp 17-139

[Simonet, 11] Simonet, A.: Automatic Production of an Operational Information System from a Domain Ontology Enriched with Behavioral Properties, Lecture Notes in Computer Science, Model and Data Engineering, 2011, Volumes 6918,pp4-17

[Tao et al., 10] Tao, F., Zhao, D., Yefa, H., Zhou, Z.: Correlation-aware resource service composition and optimal-selection in manufacturing grid, European Journal of Operational Research, Volume 201, Issue 1, 2010, pp 129-143

[Tomoaki and Ning 07] Tomoaki, S., Ning, Z.: Web Service Discovery with IOPE Using by Ontology Mapping, SIG-KBS, 2007, Volume 76, pp 87-92

[Waters et al., 09] Waters, J., Powers, B.J., Ceruti, M.B.: Global Interoperability Using Semantics, Standards, Science and Technology (GIS3T), Computer Standards & Interfaces, vol. 31, no. 6, pp 1158-1166, 2009

[Yongbo et al., 07] Yongbo, W., Xiaoying, B., Juanzi, L., Ruobo, H.: Ontology-Based Test Case Generation for Testing Web Services, International Symposium on Autonomous Decentralized Systems 2007, pp 43-50

[Zhao and Doshi, 09] Zhao, H., Doshi, P.: Towards Automated RESTful Web Service Composition, IEEE International Conference on Web Service, 2009, pp 189-196