# Developing End-User Programmable Service-Oriented Applications with VINCA

Jian Yu[1], Jianwu Wang[1,2], Yanbo Han[1], Shaohua Yang[1,2], Liyong Zhang[1,2]
[1] (Software Division, Institute of Computing Technology, CAS, Beijing 100080)
[2] (Graduate School of CAS, Beijing 100089)

## Abstract

Raising end-user's programmability is a promising way to ensure more flexible and higher-quality information provision and utilization, and to better cope with spontaneous business requirements as well. This paper presents the state-of-the-art developments of the end-user service composition language VINCA and its corresponding approach to developing end-user programmable applications on the basis of Service-Oriented Architecture. With VINCA, end-users can visually express (and later easily change) their personalized requirements from their business viewpoints, letting the underlying supporting environment take care of the technical details. The presented language and approach have been tried out with different real-world scenarios, and the evaluation in this regard is given in the paper.

## 1 Introduction

Just-in-time collaboration and virtual organization of individual applications are needed in many occasions in our networked world. The traditional way of software development, which is dependent on IT professionals, is often blocking the way due mainly to its low productivity, and thus hinders applications flexibility. End-user involvement and end-user programmability become very essential in timely development of applications. The service-oriented computing paradigm, which is currently highlighted by Web services technologies, provides an effective means of application abstraction and integration with its loosely-coupled architecture, and directly or indirectly enables end-user programming. By abstracting autonomous and heterogeneous application functionalities as services, we can assemble more flexibly individual applications to construct distributed information systems. Service composition has thus gained momentum [Bena02]. Current Web services composition languages such as BPEL4WS [Andr03] and BPML [BPMI02] are developed for IT professionals and still weak in dealing with a spectrum of application scenarios that require Web services be quickly composed and reconfigured by non-IT professionals in order to cope with the spontaneity and volatility of user requirements. Examples of such application scenarios include dynamic supply chain management, handling of city emergency, and management of massive public events [Levi03] [Reic96]. As a matter of fact, we are undertaking two real-world projects that have exactly the same requirements. The first project is called FLAME2008, which is abbreviated from *A Flexible Semantic Web Service Management Environment for the Olympic Games Beijing 2008* [Holt03]. It is an effort to develop a service mediation platform for the Olympic Games Beijing 2008, on which an effective information system providing personalized and one-stop information services to the general public, should be based. The second project is called AMGrid [Cafi02a], which targets at information sharing among different manufacturing enterprises. In fact, the problems addressed in this paper mainly come from these two projects.

In this paper, we elaborate our user-centric, business-level service composition language – VINCA shortened from *A Visual and Personalized Business-level Composition Language for Chaining Web-based Services* [Han03] and present its new developments. The core metaphor behind VINCA is: end-users can visually express (and later change) their personalized requirements from their business viewpoints, letting the underlying supporting environment take care of the technical details.

As shown in Fig. 1, the key technologies supporting VINCA are service annotation, service virtualization, service visualization and dynamic service composition[1]. Service virtualization and dynamic service composition are of major concern in this paper. With service virtualization, technical and supplementary details of Web services can be hidden and only the business-specific facets are presented to end-users. After end-users express their business requirements by composing these virtualized business-level services, a service mapping mechanism maps these virtual resources to real-world Web services. Fig. 1 shows two blocks of steps that VINCA takes to develop applications: 1) Semantic information is first added to primitive Web services, then business-level services are formed through service virtualization, finally they are visually presented to end-users through service visualization; 2) End-users "see" all available services in their business terms, and may select and compose business-level services that conform to their requirements in a just-in-time fashion. The user-composed business-level model is mapped to a software-level model while concrete Web services are dynamically bound to business-level services, and concrete Web services are invoked in interpreting the software-level model.

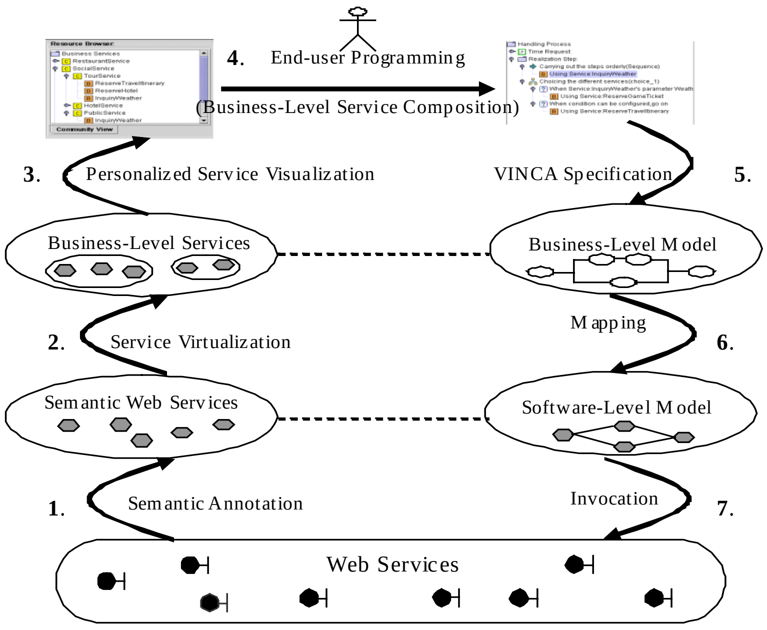


Fig. 1: Application development with VINCA

In order to achieve the above-stated end-user programmability, we have to overcome quite a large number of barriers, answering fundamental questions like: What the core programming constructs for end-users are? How they are abstracted from or related to software-level constructs? What the fundamental rules are for an end-user to grasp to program her applications? How can the end-user programming paradigm find widespread uses? This paper focuses on the first two problems and tries to resolve them with our service virtualization mechanism and business-level service composition. It is organized as follows: Section 2 presents a motivating scenario and states the

[1] We focus on a limited form of dynamic service composition. We also refer the process of composing abstract business-level services and automatically mapping these abstract services to concrete Web services as dynamic service composition.

problems VINCA wants to solve. Section 3 briefly introduces the unique features of VINCA and defines the core elements of the language. And the most important element of VINCA- Business Service is explained in detail in section 4, which reflects the core abstraction mechanism of VINCA. Section 5 analyzes the approach to applying VINCA for developing service-oriented applications, and section 6 illustrates the supporting environment of VINCA. Application and evaluation of VINCA, including a comparative study of related work, are discussed in section 7. Finally we conclude in section 8.

## 2 Problem Statement with a Motivating Scenario

To introduce the design rationales of VINCA, we choose a typical scenario from FLAME2008, which is also used as a running example in this paper.

It is assumed that, in 2008, various parties will provide a large variety of information services for the general public, and something is needed to help different groups of users (athletes, visitors, organizers, etc.) to define their personalized "applications" to make full use of the services supplied. Among the users is Mr. Johnson, a businessman on vacation. He is going to watch some games and do some sightseeing in Beijing during the Olympic Games. Before he leaves for Beijing, he can use the FLAME2008-based information system to schedule his activities and enjoy the multitude of services. Mr. Johnson plans to arrive at Beijing on Aug. 10, 2008. Since the airline must be booked one month earlier, he wants to submit his booking request at 8 AM on July 10. After booking an airline ticket successfully, he also wants to reserve a hotel and book the tennis game ticket on Aug. 12. Before setting off on Aug. 9, Mr. Johnson wishes to makes a tour reservation for the Great Wall on Aug. 11. After the tennis game at 5 PM on Aug. 12, he wants to enquire the restaurants around him, so he can conveniently find a desirable restaurant and enjoy some Chinese food. To demonstrate the volatility of requirements, let's suppose that just a few days before his setting off for Beijing, Mr. Johnson learns that it rains frequently these days in Beijing, so he decides to change his schedule. He wants to enquire the weather of Aug. 12 first and then decides his movement according to the enquired result: if it rains, he will reserve the tour to the Forbidden City, otherwise, he will reserve the tour to the Great Wall. Mr. Johnson's final schedule[2] is given in Fig. 2.

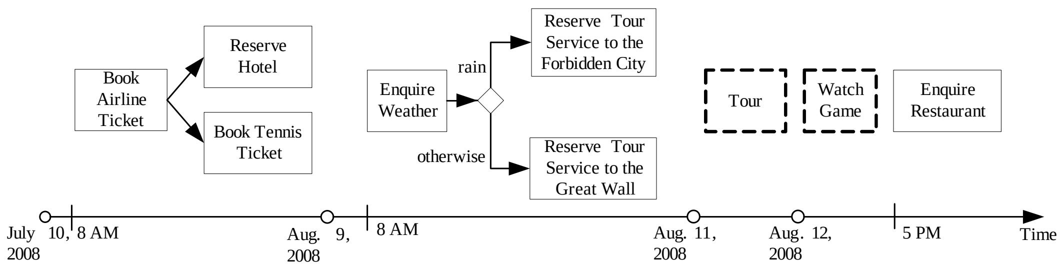


Fig. 2: Mr. Johnson's Olympic Game Schedule

Supposing that Web services that can provide above-stated functionalities (like "Book Airline Ticket", "Reserve Hotel" etc.) have already existed, to build an application that can fulfill Mr. Johnson's requirements, we can use some kind of service composition language such as BPEL or BPML. But it demands that the application builders should have IT-specific knowledge, which is difficult for normal end-users. If we can enable end-users to "program" their personalized applications by composing business-level services that they are familiar with, it will save their time and money to develop applications this way. Furthermore, it will be more convenient and

[2] The dashed rectangles denote the activities that do not need the help of any web Services.

timely for them to adjust/reconfigure their applications.

From this scenario, we can also identify the following problems:

1) Most traditional process languages only support arranging tasks by control flow, but sometimes it's more convenient for end-users to arrange their personalized services temporally.
2) Mr. Johnson is not familiar with Beijing, how can he tell "Enquire Restaurant" service his current location in a convenient way?
3) Mr. Johnson may compose his schedule on a laptop, but he may take only his PDA or smart phone with him to the tennis game, how can he still enjoy using the services?

In the next section, we discuss our solutions to the above-stated problems by introducing the basics of our end-user programmable language: VINCA.

## 3 Basics of the End-user Programming Language VINCA

### 3.1 Unique Features

Though VINCA is yet another service composition language, it differs from others in the following aspects:

#### 3.1.1 Business-level Representation of Service Resources

Fig. 3 shows the principal of VINCA metaphor. The upper right part illustrates Service Community - a supporting environment for VINCA [Cafi02b], which includes Business Services, Semantic Services, Convergent Relation and Semantic Infrastructure. Business Services are business-level services in VINCA. They are defined by domain experts to express typical domain-specific norms and functionalities, and are represented to end-users in a hierarchical structure according to some application specific classification system. Semantic Services are our semantic Web services, which are created by adding semantic annotations to Web services and registering it to Service Community. With the help of Semantic Infrastructure and Convergent Relation, functional and nonfunctional semantics of Semantic Services are captured by Business Services and further these semantics are represented to end-users visually.

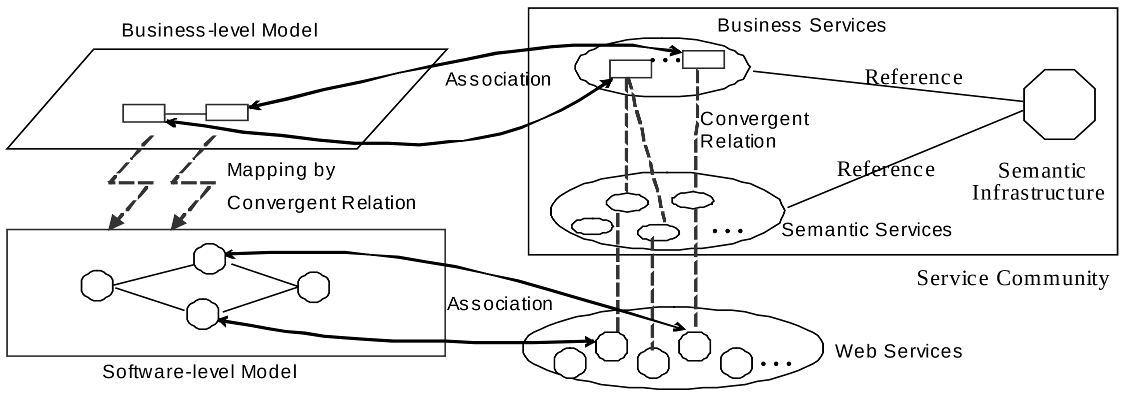


Fig. 3: The principal of VINCA metaphor

#### 3.1.2 Convergence of Business and Software Level Modeling

The concept of convergent engineering is proposed by David Taylor in 1995, who seeks to construct software system directly through business designing. It can bridge the gap between business domain and software domain and enable software to adapt to ever-changing business [Tayl95].

VINCA establishes a convergent relation to connect business-level services and software-level services. As shown in Fig. 3, a convergent relation is used to map the business-level model to the software-level model. Expressed by VINCA, a business-level model is designed to reflect business requirements with a minimal set of business-end programming concepts and mechanisms, which is easy for a business end-user to understand and master. A business end-user can (re)configure her applications by building or editing business-level model. A software-level model deals with compositions of Web Services. The Convergent Relation helps to map simple and intuitive business-level elements into more concrete and executable software-level elements and still keep the semantic consistency between them, which is supported by the Semantic Infrastructure. The Semantic Infrastructure enables the semantic references and is constructed with an ontology approach. It includes common consensus semantics and can be used by people, databases, and applications that need to share domain knowledge.

### 3.1.3 Multiple Modes of End-user Programming

As shown in Fig. 4, there are three modes of applying VINCA to compose service-oriented applications. Each mode is suitable for a certain typical usage situation and has different capability requirements for end-users.

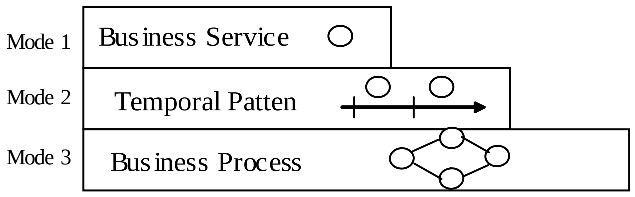


Fig. 4: Three development modes of VINCA

1) WYSIWYG (What You See Is What You Get): If an end-user can find a suitable Business Service that meets her requirements from Service Community, she can just double-click the Business Service to execute it directly. This mode has least demand on end-users' capability.
2) Semi-automatic composition with Temporal Pattern: In some situations (such as personalized application), control logic is weak and most services are executed by time in the whole process. In such situations, VINCA application can be constructed easily with Temporal Pattern. A Temporal Pattern is a block of time period that has a start time and an end time. When an end-user drops Business Services onto a Temporal Pattern, it will arrange these Business Services according to her dropping order automatically. The detailed information of Temporal Pattern can be found in [Hu04].
3) Full-fledged business programming: If there are no suitable Business Services that can directly meet an end-user's requirements and this end-users does not want to use the Temporal Pattern, she can compose her VINCA application with this mode. Firstly she can select proper Business Services from Service Community and put them onto the "Editor" field of VINCA-GUI. Then she can define the control flow of these Business Services such as sequence, parallel, switch etc. After configuring parameters for Business Services, her VINCA application is constructed. This mode is the most complex but yet the most powerful one. Users of this mode need to know some process control logic.

### 3.1.4 Awareness of User-Context

User-context plays an important role in providing personalized services to end-users. The main features include: 1) triggering service execution according to user-context; 2) providing adaptable personalized services to the user by user-context-based service selection and composition; 3) reducing interactions between the application and the user by making context as implicit inputs of the application. For example, if we treat the location information as Mr. Johnson's context and use it as an implicit input to "Enquire Restaurant" service, Mr. Johnson will always get the right information of restaurants around him without any input action. Context-aware related technologies employed by VINCA can be found in [Liu04].

### 3.1.5 Multiple Interaction Channels

Today, users have multiple choices to interact with an information system. Three kinds of interaction channels (Internet Browser, smart phone and PDA) are supported by VINCA at present to describe the different ways of service delivery. For different channels, the system will transform the response message to a suitable format, such as HTML, cHtml or WML.

## 3.2 Language Definition

In this section we concisely describe the syntax[3] of the core elements of VINCA. A complete reference can be found in Appendix 1.

### 3.2.1 VINCA Application

A VINCA application includes four parts: *vincaApplication=(businessServices, process, userContext, interaction)*. The *businessServices* defines end-user's personalized Business Services. The *process* describes the control flow and/or time order of *businessServices*. The *userContext* describes personalized user-context in a hierarchical way and the *interactions* describes the interactions between a VINCA application and its users.
The expression language of VINCA can be local or XPath [XPat99]. The local defined expression language is used in current version.

```
vincaApplication::=<vincaApplication name=”ncname” targetNamespace=”uri”
                  author=”ncname”? createDate=”date”?
                  description=”String”?
                  expressionLanguage=”local|XPath”?>
                  businessServices
                  process
                  userContext
                  interactions
                 </vincaApplication>
```

[3] The syntax definition uses an informal format to describe the XML grammar for easy reading:
- The syntax appears as an XML instance, but the values indicate the data types instead of values.
- Grammar elements in bold need further definition.
- Characters are appended to elements and attributes as follows :?(0 or 1), *(0 or more), +(1 or more).
- Elements and attributes separated by | and grouped by ( and ) are meant to be syntactic alternatives.

### 3.2.2 Business Services

The *businessServices* element is the collection of *businessService* elements. The *businessService* is the specification of end-user's personalized Business Service. An end-user's functional and non-functional requirements are captured in her personalized Business Services' semantics that include both functional semantics and QoS semantics. The *semantics* attribute is a URI linking to a DAML-S [DAML02] document that describes such semantics of a Business Service. We will discuss the details of Business Service in section 4.

### 3.2.3 Process

The syntax of *process* element is much similar to common workflow language like the ones described in [Andr03] and [Aals01]. Here we only describe the syntax of how a process node is related to a Business Service.

The *process* element consists of an *activity* element. A *task* is a generic term for a service in the process. It associates with a Business Service to be performed, also with an *interaction* element of this Business Service.

The complete process definition of Mr. Johnson's arrangement can be found in Appendix 2.

### 3.2.4 User Context

The *userContext* element records the personalized context information about an end-user. It consists of four parts: user's basic information, user's preferences and constraints including time and location. It is organized in a hierarchical structure.

#### 3.2.5 Interactions

In VINCA, a Business Service may have related interactions. Three patterns of interaction: input, output and input-output can be defined to represent the input interface, the output interface and the combined input-output interface separately. Each pattern includes a pre-defined interaction template and a channel that can be PC Internet Browser, PDA Browser or Smart Phone Browser.

## 4 The Core Abstraction Mechanism: VINCA Business Service

As stated in section 3.1.1, concrete Web services are abstracted to Semantic Services and then further abstracted to Business Services for end-users to manipulate. We establish a convergent relation between Business Services and Semantic Services to keep their semantic consistency. On the one hand, the semantics of Semantic Services are captured by Business Services with the help of convergent relation and are visually represented to end-users to achieve service virtualization. On the other hand, the convergent relation is also used to map Business Services to Semantic Services to achieve dynamic service composition. In this section, we define Business Service and the convergent relation, and discuss the usage and runtime behavior of Business Service. Finally we conclude this section by discussing the impact of Business Service to the end-user programming metaphor.

### 4.1 Business Service and Convergent Relation

As shown in Fig. 5, Business Services are outlined by domain experts according to the business specifications of a certain domain, defining typical functionalities of the domain. Semantic Services are semantically-annotated Web services that are provided by service providers. The right part in Fig.6 is the resulting construct in VINCA, it is an agglomeration of three different types of elements ($BS$, $SS$, $R_{con}$,). $BS$ is the set of all Business Services, $SS$ is the set of all Semantic Services, and $R_{con}$: $BS \leftrightarrow SS$ is the convergent relation between them. In the following sub-sections, we will define Business Service, Semantic Service and the convergent relation $R_{con}$

separately.

Apparently, not all elements at the two levels can be converged. As shown in Fig. 5, there are also Business Services (*$bs_3$*) and Semantic Services (*$ss_4$*) that can't establish a convergent relation.

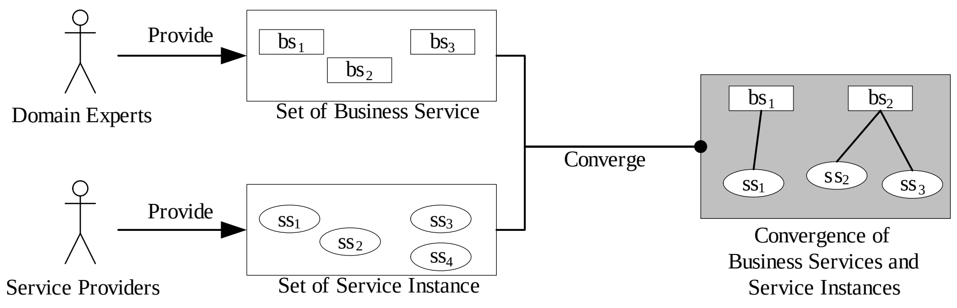


Fig. 5: Business Service, Semantic Service and their convergence

### 4.1.1 Business Service

Every Business Service *$bs_i \in BS$* is a 5-tuple: *$bs_i$= (Id, La, Au, Desc, Se)* where the first four elements represent its identity, label, author and description respectively. *Se* is its semantics and is stored in DAML-S format. *Se* is a 2-tuple: *SE= (F, QoS)*, where *F* is the functional semantics and *QoS* is the quality of service semantics. *F* is a 5-tuple, *F= (Fc, I, O, P, E)* where *Fc* is the functional classification of Business Service, I, O, P, E are inputs, outputs, preconditions and effects (we employ here the DAML-S canonical names) of Business Service in terms of ontology concepts respectively. *QoS* is the set of non-functional metric *$Q_i$*. *$Q_i$* is a 4-tuple: *$Q_i$= (Na, Op, Va, Unit)*, where *Na, Op* and *Unit* are this metric's name, comparison operator, and unit of value in terms of ontology concepts respectively. The *Va* is the value of this metric. Tab. 1 is an example of Business Service EnquireWeather.

| | |
|---|---|
| Id | EnquireWeather |
| La | Enquire Weather |
| Fc | #EnquireWeather |
| I | #Date, #Location |
| O | #WeatherCondition |
| $Q_1$ | Na: #Cost |
| $Q_2$ | Na: #Reliability |

Tab.1 Business Service EnquireWeather

Note that only the name attribute of each QoS metric of a Business Service is filled, while the other parts are left for the end-user to specify when creating her personalized Business Service.

### 4.1.2 Semantic Service

Every Semantic Service *$ss_j \in SS$* is a 7-tuple: *$ss_j$= (Id, La, Prov, Desc, WSDL-URI, Operation, Se)*, where the first six elements represent its identity, label, provider, description, related Web service WSDL file address, operation in this Web service respectively. *Se* is its semantics and has the same definition as the *Se* defined in Business Service. Tab.2 is an example of Semantic Service ForecastOfBeijing, where its functional classification #ForecastOfBeijing is subClassOf #EnquireWeather in our ontology terminology model.

| | |
|---|---|
| Id | ForecastOfBeijing |
| La | Weather Forecast of Beijing |
| WSDL-URL | http://flame:6888/Services/PublicService/ForecastOfBeijing.wsdl |
| Operation | WeatherForecast |
| Fc | #ForecastOfBeijing |
| I | #Date |
| O | #WeatherCondition |
| E | #EffectLocation(#WeatherCondition)=#LocationBeijing |
| $Q_1$ | Na: #Cost, Op: #Equal, Va: 8, Unit: #Yuan |
| $Q_2$ | Na: #Reliability, Op: #Equal, Va: 0.8 |

Tab.2 Semantic Service ForecastOfBeijing

Note that the effect semantics written in DAML-S is as follows:

```
<rdf:Property rdf:ID="effectLocation">
  <rdfs:subPropertyOf rdf:resource="&process;#effect"/>
  <rdfs:domain rdf:resource="http://flame:6888/services/concepts.daml#WeatherCondition"/>
  <rdfs:range rdf:resource="http://flame:6888/services/concepts.daml#LocationBeijing"/>
</rdf:Property>
```

### 4.1.3 Convergent Relation

A convergent relation is formed when a Semantic Service is registered to a Business Service: $Rcon = \{(bs_i, ss_j) \mid bs_i \in BS \wedge ss_j \in SS \wedge convergent(bs_i, ss_j)\}$, where the predicate $convergent(bs_i, ss_j)$ is defined as the conjunction of following formulae:

1. $Fc(ss_j)=Fc(bs_i) \vee subClassOf(Fc(ss_j), Fc(bs_i))$
2. $IO\_Match(bs_i, ss_j)$
3. $QoS\_Compatible(bs_i, ss_j)$
4. $P(bs_i)= P(ss_j)$

The first formula means that the functional classification of $ss_j$ exactly matches or is sub-class of the functional classification of $bs_i$ in the ontology hierarchy.

The second formula means that there is an I/O match between $bs_i$ and $ss_j$. We adopt the algorithm from [Paol02].

To explain formula 3, we first define the QoS compatible relation: $QoS\_Compatible(bs_i, ss_j)$ iff for all $Q_i$ of $bs_i$, there exist a $Q_j$ of $ss_j$, where $na(Q_i)=na(Q_j)$. This relation means for every QoS metric defined in Business Service, there is a counterpart in its convergent Semantic Service.

The fourth formula means that their precondition should be exactly matched.

For example, Business Service EnquireWeather and Semantic Service ForecastOfBeijing can converge, i.e., $(EnquireWeather, ForecastOfBeijing) \in R_{con}$.

## 4.2 Specification of Functional and QoS Requirements

Business Services in Service Community provides general service templates for end-users. End-users can express their functional and QoS requirements by filling in these templates and create their personalized Business Services.

To express their requirements, an end-user can drag Business Services from the classification tree and drop them onto her "Editor" field. Once a Business Service is dropped, a new personalized Business Service is created, and the end-user can further specify its functional attributes like inputs, outputs, effects etc. and QoS metrics like cost, reliability etc.
One of the novelties of our approach is that we use "Effect" semantics of Semantic Service to express the capability of Business Service and further visualize the "Effect" semantics to the VINCA graphical user interface. For example, if there is another Semantic Service ForecastOfShanghai, its semantics differ with Semantic Service ForecastOfBeijing only in the "Effect" part:
*E(ForecastOfShanghai): #EffectLocation(#WeatherCondition)=#LocationShanghai*
So when ForecastOfBeijing and ForecastOfShanghai are both registered to Business Service EnquireWeather, end-users will know the capability of EnquireWeather, and they can specify their functional requirements within the capability range. Note that the capability of a Business Service is dynamic for the dynamism of its registered Semantic Services.
For example, Tab.3 is Mr. Johnson's EnquireWeather Business Service. He wants to know the weather of Beijing and the cost of this service should not exceed 10 Yuan.

| | |
|---|---|
| Id | EnquireWeather_Johnson |
| La | Enquire Weather |
| FC | #EnquireWeather |
| I | #Date, #Location |
| O | #WeatherCondition |
| E | #EffectLocation(#WeatherCondition)=#LocationBeijing |
| $Q_1$ | #Cost, #LessThan, 10, #Yuan |

Tab.3 Mr. Johnson's personalized Business Service according to his requirements

## 4.3 Runtime Behavior of Business Service

To "execute" a Business Service at runtime, it should be mapped to Semantic Services first. A matching algorithm is used to select elements among the Semantic Services that are registered to it. In addition, if there are many candidates, a ranking algorithm adopted from [Siva03] is used.
The matching algorithm follows the criteria described bellow:
1. *EMatch($ss_j$ ,$bs_i$)*
2. *QoS_Satisfiable($bs_i$, $ss_j$)*

In our current approach, *EMatch($ss_j$ ,$bs_i$)* is defined as *E($ss_j$)= E($bs_i$)*. So there should be an exact matching between the effect of $ss_j$ and the effect of $bs_i$. But we are working on an approach to express *EMatch* relation as logically entailment [Nils98]. For example, if some Semantic Service can forecast the weather of Beijing and Shanghai, it can also match in effect with the Business Service EnquireWeather_Johnson in Tab. 3, for its capability can satisfy the needs of EnquireWeather_Johnson.
We convert the QoS_Satisfiable relation to a constraint satisfaction problem [Marr98]. It is a normal approach and we do not discuss it here.

We have discussed the internal structure and various semantics of Business Service in the above sub-sections. Here we discuss why these constructs can support end-user programming.
First of all, Business Services are virtualized representation of concrete Web services. From its definition, we can see that technical details of Web services such as WSDL address, concrete data

types are hidden. We also can see from convergent relation that one Business Service represents the aggregation of a group of functional related Web services. The functional and non-functional semantics of these Web services are captured by Business Service through convergent relation. Secondly, Business Services are organized according to their well-defined functional classification semantics. So end-users have unanimous understanding to a concept and can locate their needed Business Services easily. Thirdly, with the help of “Effect” semantics, end-users can know the real capability of a Business Service. By specifying their personalized Business Services according to their requirements, the matching algorithm will dynamically select the proper Web services to execute for end-users.

## 5 An Approach to Applying VINCA for Service-Oriented Applications

The process of applying VINCA to compose service-oriented applications can be divided into three stages: 1) Establish Service Community – the host of Business Services, Semantic Services and Convergent Relation and Semantic Infrastructure, or use an already-existed one. 2) End-users compose their programs visually by drag-and-dropping Business Services, connecting them with control constructs or put them into the time slices, and then configuring their parameters like inputs/outputs, interactions, context etc. 3) End-uses execute their programs, and interact with them or monitor their running status. We will explain these three stages respectively in the following subsections through the scenario illustrated in section 2.

### 5.1 Prerequisite – Building-Up of Service Communities

End-users compose VINCA programs on the basis of an existing Service Community. Service Community organizes business services with an application-specific classification system for the purpose of navigation and maintenance. The content of service community can be filled in the following way: Domain experts establish the semantic infrastructure that can provide the ontology for concepts in the application domain so that different users of the same domain can have a unanimous understanding of concepts. Utilizing this pre-defined ontology, domain experts define the specifications of Business Services that represent typical user requirements in this domain. Still with the help of semantic infrastructure, service providers can add semantic information to concrete Web services to create Semantic Services. Then service provider will add them to Service Community and try to register them to Business Services. But Service Community will check the semantic consistency, and the convergent relation may not be established if the check fails.

Let’s take the scenario illustrated in section 2 as an example: “Weather”, “Ticket”, “Hotel” and other basic concepts should be described firstly to establish the semantic infrastructure in the Service Community; after analyzing the typical user requirements in the FLAME2008 project, “Enquire Weather”, “Book Plane Ticket”, “Book Hotel” and other Business Services are defined; then semantic Web services such as forecastOfBeijing and forecastOfShanghai are added to the Service Community and the convergent relation between Business Service EnquireWeather and them will be established.

### 5.2 End-User Programming

As stated in section 3.3, end-users can construct their VINCA applications in different ways. Users can also configure the context and interactions visually.

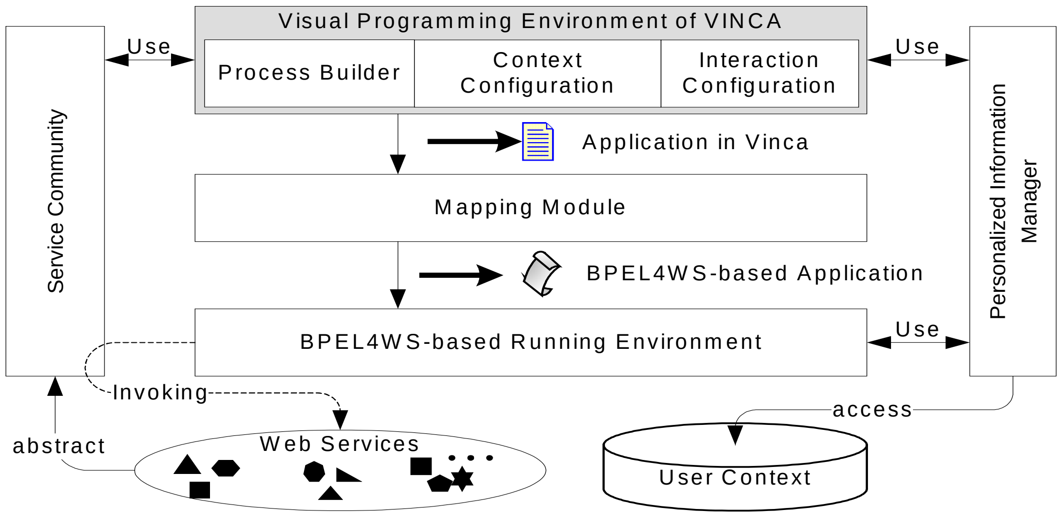
According to the scenario illustrated in section 2, Mr. Johnson may arrange his schedule with the help of Temporal Pattern. He can drag-and-drop desired Business Services into proper time slices, and then further can arrange them with control logic if needed. He may also configure the "location" input parameter of "Enquire Restaurant" Business Service with his location context and the interaction mode as "PDA" so that he can enquire the restaurant around him at dinnertime.

### 5.3 Just-in-Time Execution

After Mr. Johnson finished the arrangement of his schedule, he can monitor the execution state of his VINCA application and interact with it conveniently. For example, when the "Enquire Restaurant" Business Service is executed, the instant location of Mr. Johnson can be retrieved from his PDA and be set as the input parameter. After the execution of the "Enquire Restaurant" Business Service, the information of restaurant around him can be sent to his PDA.

## 6 A Supporting Environment: the VINCA Studio

We have implemented a prototype system called VINCA Studio to support end-user building Web-based applications with VINCA. The system architecture of VINCA is shown in Fig. 6.



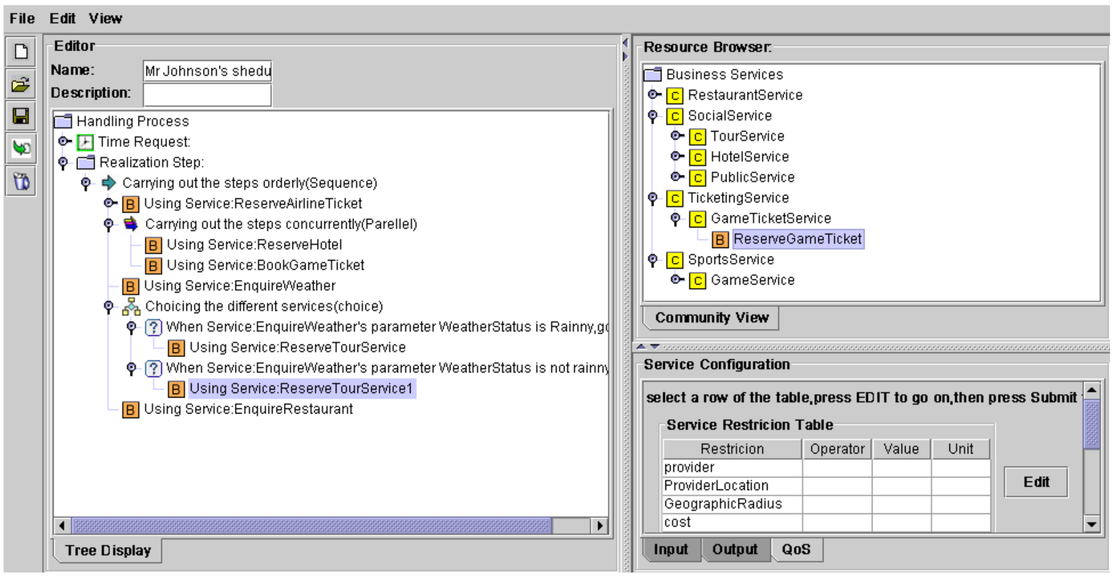
Visual Programming Environment of VINCA
Process Builder
Context Configuration
Interaction Configuration
Use
Use
Service Community
Personalized Information Manager
Application in Vinca
Mapping Module
BPEL4WS-based Application
BPEL4WS-based Running Environment
Use
Invoking
access
abstract
Web Services
User Context

Fig. 6: the Supporting Environment: the VINCA Studio

The VINCA Visual Programming Environment enables end-users to express their requirements from business viewpoint in a WYSIWYG (What You See Is What You Get) manner. Its output is the so-called VINCA program – a set of specifications in XML. Then, the VINCA Mapping Module will transform them into BPEL4WS-alike executable specifications.

The BPEL4WS-based Running Environment is responsible for chaining Web-based services at operational level, and binding and invoking individual services. It consists of event manager, application scheduler, location manager, time manager and process engine.

Fig. 7 is a snapshot of an end-user's working interface supported by the business-end programming environment that includes three components: Resource Browser, Application Editor Panel, and Service Configuration Panel. The Resource Browser renders the well-organized Business Services that can be used in public services; end-users can drag the needed Business Services and drop them to the Application Editor Panel to compose their personalized application. Through the Service Configuration Component, users' personalized requirements on a Business Service can be set.



Name: Mr Johnson's shedu
Description:

```
Handling Process
  Time Request:
  Realization Step:
    Carrying out the steps orderly(Sequence)
      Using Service:ReserveAirlineTicket
      Carrying out the steps concurrently(Parellel)
        Using Service:ReserveHotel
        Using Service:BookGameTicket
      Using Service:EnquireWeather
      Choicing the different services(choice)
        When Service:EnquireWeather's parameter WeatherStatus is Rainny,g
          Using Service:ReserveTourService
        When Service:EnquireWeather's parameter WeatherStatus is not rainny
          Using Service:ReserveTourService1
      Using Service:EnquireRestaurant
```

```
Business Services
  RestaurantService
  SocialService
    TourService
    HotelService
    PublicService
  TicketingService
    GameTicketService
      ReserveGameTicket
  SportsService
    GameService
```

select a row of the table,press EDIT to go on,then press Submit

Service Restricion Table

| Restriction | Operator | Value | Unit |
|---|---|---|---|
| provider | | | |
| ProviderLocation | | | |
| GeographicRadius | | | |
| cost | | | |

Fig. 7: Snap-shot of VINCA Programming Environment

## 7 Application and Evaluation of VINCA

VINCA has been applied in several projects including FLAME2008 and AMGrid. AMGrid has been shipped to its end-users in July 2004. Tab. 4 shows some statistics sampled from part of the AMGrid project. Comparing with the end-user programming time, much more time is spent on constructing a user-friendly AMGrid Service Community (including Web services Development, Ontology Building and Business Service Definition). But the process of building the Service Community is a one-time effort. Little time is needed to maintain it as soon as it is built. We can also find that the time of composing a business process has been shortened to a magnitude of minutes. Furthermore, it will be easy for end-users to reconfigure their processes, relieving the burden of software maintainers. It is a very promising comparing to the situation of traditional software development: software developers may delve themselves into coping with infrastructure-level requirements, and end-users will be satisfied with composing and adjusting their own business processes freely.

| Service Community Scale | | Service Community Build Time | | | | EUP Time |
|---|---|---|---|---|---|---|
| Business Service Quantity | Semantic Service Quantity | Building Domain Ontology | Defining Business Services | Developing Web services | Defining Semantic Service | Average Time for end-user to compose a process |
| 20 | 30 | 4 Man*Day | 10 Man*Day | 18 Man*Day | 2 Man*Day | 20 Minute |

Tab. 4: AMGrid Project Statistics

Our work relates to several fields of research and technologies, including service virtualization, dynamic service composition and context-aware computing.

We focus on the virtualization of a group of functional related Web services. Service Domain [Khal03] [Tan03] represents a collection of comparable or related Web services through a common services entry point. Service Container [Bena03] aggregates several other substitutable services that provide a common capability. Although VINCA Business Service also represents a group of capability related services, the main difference between these work and VINCA lies on the user accessibility of VINCA Business Service.

We achieve dynamic service composition by mapping abstract process to concrete BPEL4WS process. [Mand03] presents an approach to combining DAML-S and BPEL4WS for achieving dynamic binding, where user-defined constraints are used in service selection. [Siva04] developed a template-based approach to capturing the semantic requirements of process services in the METEOR-S Web Service Composition Framework. [Akki04] provides a prototype workflow engine that accepts abstract BPEL4WS flows augmented with semantic annotations in DAML-S and performs runtime discovery, composition, binding and execution of web Services. A significant difference between the existing efforts and our approach lies in that VINCA further presents the semantic information of Web services to end-users visually. Mapping from abstract Business Services to concrete Web services is done by selecting appropriate semantic Web services from our Service Community. Such mapping does not need a service discovery process. In VINCA, context-awareness is also considered to some extent.

## 8 Conclusion

In this paper, we have presented an end-user programmable dynamic service composition language VINCA. The most significant feature of this language is its end-user programmability. By virtualizing service resources to a business-level while keeping their semantic consistent with software-level services, presenting multiple programming modes and sensing the user context, end-users can “program” many of their applications with VINCA.

The development of VINCA is guided by a number of real-world scenarios to highlight its practical usefulness. VINCA is used in the FLAME2008 and AMGrid projects to allow business experts to transparently examine all scattered resources that are accessible and available to them and to configure and reconfigure these resources in an easy and straightforward manner.

In the future, we plan to design a mechanism that allows end-users to express their requirements at larger granularity abstraction level: a business process composed by Business Services can be abstracted as another Business Services, and so on. The research work on a more effective algorithm to implement the “Effect” semantics matching is ongoing.

## References

[Aals001] WMP van der Aalst, A. Kumar, XML based schema definition for support of Inter-organizational workflow, University of Colorado and Eindhoven University of Technology Report, http://spot.colorado.edu/ ~akhil/pubs.html, 2001.

[Akki04]Rama Akkiraju, Kunal Verma, Richard Goodwin1, Prashant Doshi, Juhnyoung Lee, Executing Abstract Web Process Flows, The 14th International Conference on Automated Planning and Scheduling , (ICAPS 2004), Whistler, British Columbia, Canada, June 3-7 2004.

[Andr03] T Andrews, F Curbera, H Dholakia, et al., Business Process Execution Language for Web Services, 2003. http://www-106.ibm.com/developerworks/ webservices/library/ws-bpel/

[Bena02] Benatallah B., Dumas M., Fauvet M. C. and Rabhi F.A, Towards Patterns of Web Services Composition, Patterns and Skeletons for Parallel and Distributed Computing, S. Gorlatch and F. Rabhi (Eds), Springer Verlag (UK) 2002.

[Bena03] B Benatallah, Q Sheng, M Dumas, The Self-Serv Environment for Web Services Composition, IEEE Internet Computing, pp. 40-48, 2003, 7(1).

[BPMI02] BPMI.org, Business Process Modeling Language, http://www.bpmi.org/, 2002.

[Cafi02a] CAFISE group, AMGrid Project, Technical Report, Software Division, ICT, CAS, 2002.

[Cafi02b] CAFISE group, Service Community Specification, Technical Report, Software Division, ICT of CAS, December 2002.

[DAML02] The DAML Service Coalition, DAML-S: Semantic Markup for Web Services, http://www.daml.org/services/, October 2002.

[Doshi03] Prashant Doshi, Richard Goodwin, Rama Akkiraju. Parameterized Semantic Matching for Workflow Composition. Draft In the works. Still to be published.

[Han03] Y Han, H Geng, H Li et al, VINCA - A Visual and Personalized Business-level Composition Language for Chaining Web-based Services, First International Conference on Service-Oriented Computing, Trento, Italy, pp. 165-177, 2003.

[Holt03] B. Holtkamp, R. Gartmann, Y. Han, FLAME2008-Personalized Web Services for the Olympic Games 2008 in Beijing, Proceedings of eChallenges 2003, Bologna, Italy, Oct. 2003.

[Hu04] Haitao Hu ,Yanbo Han,Kui Huang , Gang Li, Zhuofeng Zhao, A Pattern-based Approach to Facilitating Service Composition, The Third International Workshop on Grid and Cooperative Computing (GCC2004), Wuhan, China, 2004.

[Kei02] Keith Levi, Ali Arsanjani, A Goal-driven Approach to Enterprise Component Identification and Specification, Communications of the ACM, p. 45-52, 2002. 45(10).

[Kha03] R Khalaf, F Leymann, On Web Services Aggregation, 4th International Workshops on Technologies for E-services in Conjunction with the VLDB conference, Berlin, Germany, pp. 1-13, 2003.

[Levi 2003] D.S.Levi et al., Designing and Managing the Supply Chain, McGraw-Hill, 2003.

[Liu04] Hao Liu, Yanbo Han, Gang Li and Cheng Zhang, Achieving Context Sensitivity of Service-oriented Applications with the Business-end Programming Language VINCA, The Third International Workshop on Grid and Cooperative Computing (GCC2004), Wuhan, China, 2004.

[Mand03] Mandel, D., McIlraith S., 2003 Adapting PBEL4WS for the semantic web: The bottom up approach to web service interoperation *Second International Semantic Web Conference (ISWC2003),* Sanibel Island, Florida, 2003.

[Marr98] K Marriot, P Stuckey. Programming with Constraints: An Instruction. MIT Press, 1998.

[Nils98] Nils J. Nilsson, Artificial Intelligence, A New Synthesis, Morgan Kaufmann Publishers, chapter 13.8.2-metatheorems, 1998.

[Paol02] Paolucci, M., Kawamura, T., Payne, T., and Sycara, K., Semantic matching of web services capabilities, *International Semantic Web Conference, Sardinia, Italy, June 9-12, 2002*.

[Reic96] Reichert, M., Kuhn, K., and Dadam, P, Process Reengineering and Process Automation in Clinical Application Environments (in German), Proc. GMDS'96 (pp. 219-223). Bonn, Germany: MMV Medizin Verlag, 1996.

[Siva03] Sivashanmugam, K. , The METEOR-S Framework for Semantic Web Process Composition , M.S. Thesis , Department of Computer Science, University of Georgia, Athens, GA.

Slides: pdf , powerpoint-show, 2003.

[Siva04] Kaarthik Sivashanmugam, John Miller, Amit Sheth, and Kunal Verma, Framework for Semantic Web Process Composition, Semantic Web Services and Their Role in Enterprise Application Integration and E-Commerce, Special Issue of the International Journal of Electronic Commerce (IJEC), Eds: Christoph Bussler, Dieter Fensel, Norman Sadeh, Feb 2004.

[SOA] www.service-architecture.com

[Tayl95] D. Taylor, Business Engineering with Object Technology, John Wiley & Sons, 1995.

[Tan03] Y Tan, B Topol, V Vellanki, et al, Business service grid: Manage Web services and Grid services with Service Domain Technology, 2003. http://www-106.ibm.com/developerworks/grid/library/gr-servicegrid

[XPath, 1999] XML Path Language (XPath)Version 1.0, http://www.w3.org/TR/1999/REC-xpath-19991116

## Appendix 1. VINCA Syntax

1. VINCA Application

**vincaApplication**::=<vincaApplication name=”ncname” targetNamespace=”uri”
    author=”ncname”? createDate=”date”?
    description=”String”?
    expressionLanguage=”local|XPath”?>
    **businessServices**
    **process**
    **userContext**
    **interactions**
    </vincaApplication>

2. Business Service

**businessServices**::=<businessServices>
    <businessService name=”ncname” label=”String”
    author=”String”? description=”String”?
    semantics=”anyURI”?/>+
    </businessServices>

3. process

**process**::=<process name=”ncname”?>
    **activity**
    </process>

4. activity

**activity**::=**task** | **sequence** | **switch** | **parallel** | **while** | **wait** | **schedule** | **terminate**

5. task

**task**::=<task name=”ncname” BusinessService=”ncname”
    interaction=”ncname”
    start_time=”date”? end_time=”date”?/>

6. sequence

**sequence**::=<sequence>
    **activity**+

</sequence>

7. switch

**switch**::=<switch>
<case condition="**bool-expr**">+
**activity**
</case>
<otherwise>?
**activity**
</otherwise>
</switch>

8. parallel

**parallel**::=<parallel>
**activity**+
</parallel>

9. while

**while**::=<while condition="bool-expr">
**activity**
</while>

10. wait

**wait**::=<wait for="**duration-expr**" | until="**deadline-expr**" />

11. schedule

**schedule**::=<schedule from="dateTime" to="dateTime">
**activity**+
</schedule>

12. terminate

**terminate**::=<terminate/>

13. context

**userContext**::=<userContext name="ncname" admin="ncname">
**identity**
**preference**
**location**
**time**
</userContext>

14. identity

**identity**::=<identity>
<identityItem>*
**contextItem**
</identityItem>
</identity>

15. preference

**preference**::=<preference>
<preferenceItem>*
**contextItem**
</preferenceItem>
</preference>

16. location

**location**::=<location>
                  **contextItem**
                </location>

17.    time

**time**::=<time>
            **contextItem**
          </time>

18.    contextItem

**contextItem**::=<contextItem category=”String” semantics=”anyURL” value=”String”/>

19.    interactions

**interactions**::=<interactions>
                        <interaction name=”ncname”>
                            <pattern mode=”input|output|inout”>+
                               <template name=”anyURI”/>
                               <channel name=”PC|PDA|SmartPhone”/>
                            </pattern>
                        </interaction>
                     </interactions>

## Appendix 2. Process Definition of Mr. Johnson’s Schedule

```
<?xml version="1.0"?>
<process name="ArrangementOfJohnson">
  <sequence>
    <schedule from="2008-07-10T08:00:00" to="2008-07-11T00:00:00">
       <sequence>
           <task name="bookPlaneTicket" businessService="bookPlaneTicket"
                    interaction="interactionBPT"/>
           <parallel>
              <task name="reserveHotel" businessService="reserveHotel"
                        interaction="interactionRH"/>
               <task name="bookTennisTicket" businessService="bookOlympicGameTicket"
                        interaction="interactionBOGT"/>
           </parallel>
        </sequence>
    </schedule>
    <schedule from="2008-08-09T08:00:00" to="2008-08-10T00:00:00">
        <sequence>
        <task name="enquireWeather" businessService="enquireWeather"
                 interaction="interactionEQW"/>
        <switch>
           <case condition="enquireWeather.output.WeatherCondition=&concept;#rainy">
             <task name="reserveTour" businessService="reserveTour "
                      interaction="interactionRESTOUR"/>
           </case>
           <otherwise>
```

```
                <task name="reserveTour" businessService="reserveTour "
                         interaction="interactionRESTOUR"/>
              </otherwise>
           </switch>
         </sequence>
     </schedule>
      <schedule from="2008-08-12T17:00:00" to="2008-08-12T17:30:00">
          <task name="enquireRestaurant" businessService="enquireRestaurant"
                    interaction="interactionENQRESTR"/>
      </schedule>
    </sequence>
</process>
```