An Interactive Service Customization Model

Jian Cao[a*], Jie Wang[b], Kincho Law[b], Shensheng Zhang[a], Minglu Li[a]

[a] Department of Computer Science, Shanghai Jiaotong University,
200030, Shanghai, P. R. China

[b] Department of Civil and Environment Engineering, Stanford University,
Stanford, CA 94305, U.S.A

Abstract: Mass customization has become one of the key strategies for a service provider to differentiate itself from its competitors in a highly segmented global service market. This paper proposes an interactive service customization model to support individual service offering for customers. In this model, not only that the content of an activity is customizable, but the process model can also be constructed dynamically according to the customer's requirements. Based on goal ontology, the on-demand customer requirements are transformed into a high-level service process model. Process components, which are building blocks for reusable standardized service processes, are designed to support on-demand process composition. The customer can incrementally define the customized service process through a series of operations, including activation of goal decomposition, reusable component selection, and process composition. In this paper, we first discuss the key requirements of the service customization problem. We then present in detail a knowledge based customizable service process model and the accompanying customization method. Finally we demonstrate the feasibility of the our approach through a case study of the well-known travel planning problem and present a prototype system that enables users to interactively organize a satisfying travel plan.

Keywords: Service Customization; Service Process; Goal Ontology; Process Component

1. Introduction

Internet has grown beyond being an information-sharing platform and is fast becoming a business transaction platform by providing resource-sharing functions through the interactions between the consumers and the providers of Web services. From the IT perspective, a Web service is a kind of self-described and self-contained component that can be discovered and invoked to provide certain functions through the network. Web service technologies, such as WSDL (W3C, 2001), UDDI (OASIS, 2002) and SOAP (W3C, 2003), are among the most active research topics both in academia and business areas. The business transaction model based on Web services challenges the assumption that the Internet is just as an additional channel, and it's impact on business is mainly to increase the speed of existing production processes (Giacomo, 2001).

The Web services offered by the Internet can bring profits to service providers, in a way very similar to traditional service companies. As a provider of services, albeit through Internet or other traditional channels, business strategy is the key to success. One important business strategy is mass customization that can potentially differentiate one company from others in a highly competitive and segmented market. Mass customization, which was originated in marketing, requires providing a customer with customized products and services but without exceeding the

*Corresponding Author. Tel.: 86-21-62933536; Fax.: 86-21-62933536; Email: cao-jian@cs.sjtu.edu.cn
Address: Department of Computer Science &Technology, Shanghai Jiaotong University, 1954, Huashan Road, Shanghai, 200030, P.R.China

price of comparable standard products (Duray et al., 2000). Although mass customization may appear to be mainly for the manufacturing enterprise for delivering manufactured products to customers, this business strategy is also important for service providers. For example, a research conducted by IBM Malaysia's services department shows that there are three important factors affecting on their adoption of mass customization (Perters and Saidin, 2000):

(1) Heterogeneity of market demands
(2) Customers' demands of fast and varied response to their needs
(3) Competition from other related enterprise

Furthermore, it is predicted that in the next few years mass customized services will generate more revenue than either one-of-a-kind or on-the-rack services. Obviously, Web service providers will face similar challenges. The basic requirements to provide mass customization capability in service offering include supports for flexible business processes, organizational structures and enterprise resources. This paper focuses on the development of an interactive service customization model that can enhance the flexibility of Web service providing process.

Current technologies support three basic ways of using Web service, as illustrated in Figure 1:

1) Individual Web Service Invocation (Figure 1 (a))

Some standards such as WSUI (WSUI Working Group, 2002) provide the interface through which a customer can make use of a Web service directly. When a single Web service does not meet the requirements of the customer, manual efforts are needed to compose and coordinate multiple Web services, which could be difficult if not impossible for the customer. This model may work for simple applications, for example, weather forecast or map services.

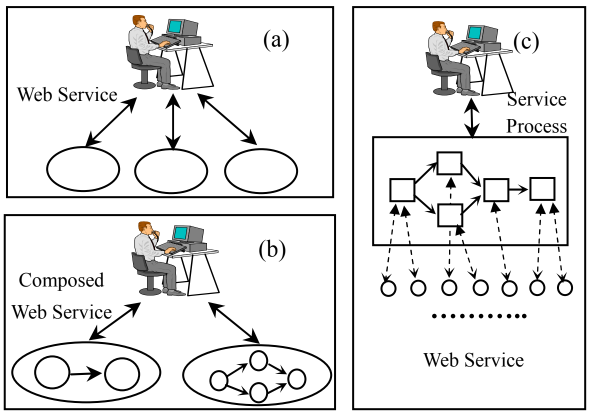


Fig. 1. Different Interaction Modes between a Customer and Services

2) Composed Web Services Invocation (Figure 1 (b))

It is possible to compose multiple Web services to provide a high level service to the customer. Typically, composed Web services and their relationships are explicitly defined in the high level service for specific applications (Srivastava and Koehler, 2003). The process and services are rigidly defined and are difficult to customize.

3) Service Process Invocation (Figure 1 (c))

Another alternative is to define a service process model consisting of activities and the requirements of the supporting Web services (Sivashanmugam et al, 2003). Instead of binding Web services to the activities statically, only the requirements for the Web services are defined as activities' contents. Appropriate Web services are discovered and invoked dynamically during run time. In this model, although the Web services are selected according to the requirements the structure of the process is predefined and only limited customization capabilities are allowed.

The objective of this work is to develop a process and service model that has the flexibility to support mass customization. The proposed service customization model allows customization of not only the content of the activity, but also the structure of the service process, which is built dynamically according to the customers' requirements. Based on goal ontology, the customer's requirements are transformed into an abstract service process in which the process components, which are some reusable standardized service processes, serve as building blocks to support process composition. The customer can define the entire customized service process through goal decomposition, process composition and process component reuse.

This paper is organized as follows. Section 2 discusses the key requirements of the service customization problem. Section 3 presents a knowledge based customizable service process model. Customization of a service is discussed in details in Section 4. Section 5 describes a demonstration system. Section 6 discusses related works and, finally, Section 7 briefly summarizes the paper and points out several directions for future works.

2. Basic Requirements

We use travel plan as a case study to delineate some of the basic requirements for service customization. The travel plan has been used as a case study by many researchers (Orriëns et al., 2003; Srivastava and Koehler, 2003). Although the travel plan problem appears to be relatively simple, it contains most of the perspectives of a service process and reveals many difficulties in the service customization problem.

Several tasks are involved when making a travel plan. First, a customer needs to decide on how to go to the destination and return from there and select the means for transportation. The customer may then need to reserve for a hotel and, occasionally, book a taxi or a shuttle bus for transportation from the airport (or train station) to the hotel or from the hotel to the airport (or train station). At first glance, the planning process may appear to be quite simple, however, multiple scenarios could happen. For example, the customer can choose to travel by train or by airplane. If by airplane, the customer needs to select a flight. Given a specific flight, there are also multiple ways to book the ticket. For example, the customer may prefer to book the flight directly or search and inquire for the most suitable schedule. For a specific travel plan, hotel reservation may depend on overnight stay or other conditions, such as locations and convenience. In addition, foreign travels may also require a visa. In short, many alternatives exist and decisions are involved when planning for a travel.

One important observation in the travel plan problem is how many of these variations are not known a priori to the service provider at the beginning of service offering. The challenge is to design an end-user friendly information system that can satisfy customers' requirements. The most important requirements are:

1) Satisfy diverse customer requirements with predefined system models

To develop a service process model that can meet all kinds of customer's needs is difficult if not impossible. There are two basic approaches to develop a general framework to handle the diverse various service processes. The first approach is based on generalization and the other is to integrate and compose small models. Generalization provides abstract process models to support a broad range of scenarios. An abstract process model can be transformed to support useful scenarios through specialization. General process patterns (abstract process models) exist within a domain as well as across different domains (Malone, 2003). For example, when making reservation for a train ticket or a flight ticket, the customer may need to inquire about the ticket information, select one and then book it. In addition to generalization, a process model should support composition such that individual processes can compose a more complex process model.

That is, a set of reusable process unit models should be maintained. Generalization and composition provide many possibilities to satisfy diverse customer's requirements while keeping the technical complexity at an acceptable level.

2) Integrate system knowledge and customer's personal knowledge

Providing services is a knowledge intensive task. It must contain different levels of organized knowledge about a specific domain. For example, while a ticket booking process model must include the knowledge about booking a ticket, different ticket booking process models with knowledge about alternative solutions to satisfy different requirements must also exist. From the system perspectives, these process models contain explicit knowledge about ticket reservation. However, there is also implicit knowledge that stay in a customer's mind, which is difficult to model in advance. For example, should a customer reserve first a departure ticket or a return ticket? This scenario depends on the need of the customer. Whether the customer should rent a car or book a taxi at the destination airport is also implicit information that may not be known in advance. Therefore, the IT system framework must provide a dynamically means to integrate system knowledge and customer's personal knowledge.

3) Allow dynamic refinement on process model

In order to make the process model flexible and customizable, the relationships among the activities of a reusable abstract process model should be loosely defined. The customer should be allowed to incrementally add dependencies to the activities of an executing process model. For example, the customer may add a new ordering relationship between the activities for reserving a departure ticket and reserving a return ticket. The customer may also want to connect the data from one activity to another, for example to link the results of reserving a departure ticket to the activity for booking a car rental. Sub-processes may also be included in the model on demand. For example, upon finding that booking a foreign flight requires a visa, a sub-process of visa application should be added to the process. Flexible process representation and execution are needed to support incremental refinements of the executing process model.

The service customization model described in the paper attempts to address these three basic requirements.

3. A Knowledge-based Customizable Service Process Model

To handle diverse customers requirements, a knowledge-based customizable service process model, which is composed of domain ontology, goal ontology and process components as depicted in Figure 2, is proposed.

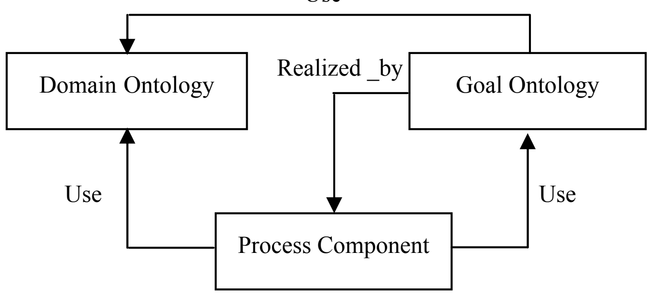


Fig. 2. A Knowledge-based Customizable Service Process Model

3.1 Domain Ontology

In order to provide an interface with unambiguous and consistent representations to the customers and to support reasoning within the service process customization, the concepts employed in the service process model should be formalized and be shared across the sub-models. Formal ontology is one approach to specify content-specific agreements for a variety of knowledge-sharing activities. Ontology is an explicit specification of a conceptualization and its importance has been well recognized (Gruber, 1993; Guarino, 1998). Domain ontology defines a set of concepts and their formats for a specific domain application. For example, Figure 3 shows parts of domain ontology for a travel plan problem, where concepts include ticket, time and hotel etc. The domain ontology is structured as a set of individual generalization hierarchy terminology trees, with the more abstract concepts of the ontology forming the root terms of which other terms are specified. Each term of the hierarchy may be associated with a number of named attributes. Attributes are specified with an attribute name and type. Examples of built-in primitive types include Boolean, string, byte, integer, and real number. The complex types can be terms defined in other term trees. Attributes of a term are inherited by all of its children, which may have additional attributes.

If a term a is inherited from a term b, a specializes b and it is denoted as $a \in SP_t(b)$. Accordingly, b generalizes a and it is denoted as $b \in GE_t(a)$. The relationships of specialization (or generalization) are transitive. For example, if $c \in SP_t(a)$ and $a \in SP_t(b)$, then $c \in SP_t(b)$.

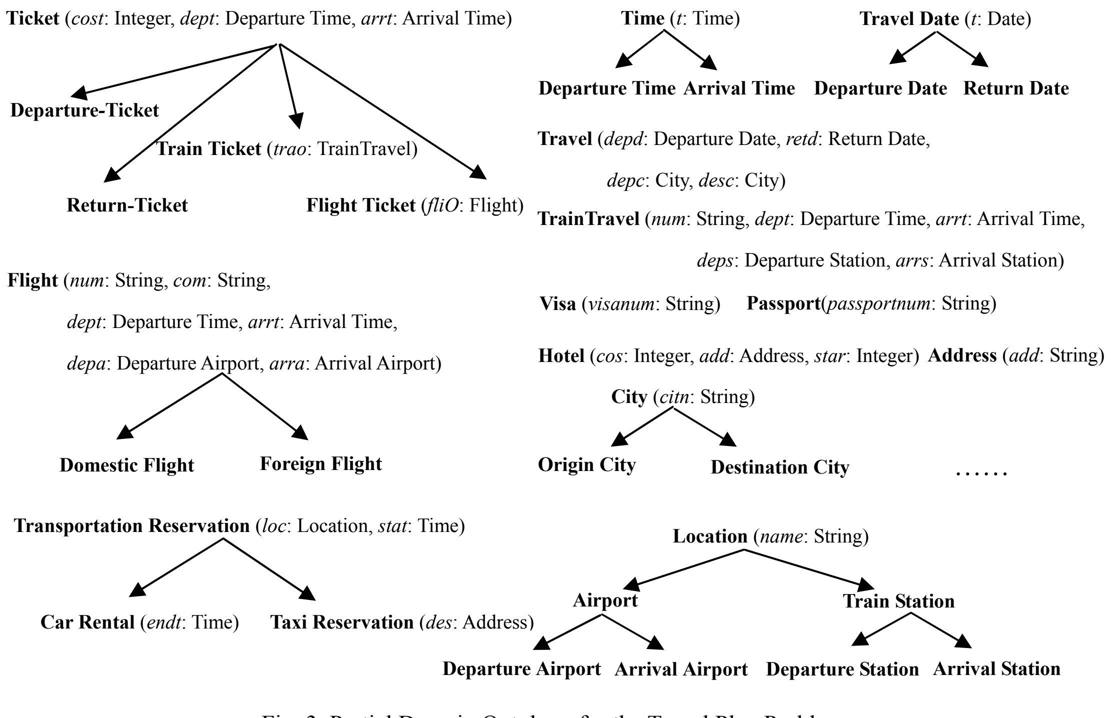

Ticket (cost: Integer, dept: Departure Time, arrt: Arrival Time)
Departure-Ticket
Train Ticket (trao: TrainTravel)
Return-Ticket
Flight Ticket (fliO: Flight)
Time (t: Time)
Departure Time
Arrival Time
Travel Date (t: Date)
Departure Date
Return Date
Travel (depd: Departure Date, retd: Return Date, depc: City, desc: City)
TrainTravel (num: String, dept: Departure Time, arrt: Arrival Time, deps: Departure Station, arrs: Arrival Station)
Flight (num: String, com: String, dept: Departure Time, arrt: Arrival Time, depa: Departure Airport, arra: Arrival Airport)
Domestic Flight
Foreign Flight
Visa (visanum: String)
Passport(passportnum: String)
Hotel (cos: Integer, add: Address, star: Integer)
Address (add: String)
City (citn: String)
Origin City
Destination City
……
Transportation Reservation (loc: Location, stat: Time)
Car Rental (endt: Time)
Taxi Reservation (des: Address)
Location (name: String)
Airport
Train Station
Departure Airport
Arrival Airport
Departure Station
Arrival Station

Fig. 3. Partial Domain Ontology for the Travel Plan Problem

3.2 Goal Ontology

In order to allow the customers customizing their service processes, goal ontology is provided as a high level knowledge.

Human action follows specific goals, i.e., targets, intentions and purposes, for their activities. Goal is a high level concept and the relationships among goals represent the domain knowledge. A specific goal can be achieved by different methods. The concept of goal has different meanings in different contexts. In requirement engineering, a goal is defined as "something that some stakeholder hopes to achieve in the future "(Rolland and Ben, 1998). On the other hand, if we design and implement a system, the goal would be to offer certain services by using this system. The former is called a requirement goal and the latter an operation goal. A requirement goal *g* is achieved by using those methods that have operation goals supporting *g*.

Rolland et al. formalized a goal structure for requirements engineering (Rolland et al., 1998). In their structure, a goal has a verb and a set of parameters. These parameters include target, direction, way, beneficiary, referent, quality, location and time. Although the goal structure is well defined, some of the parameters such as direction, beneficiary and referent are not applicable to the service-providing domain.

In the proposed goal ontology model, depicted in Figure 4, a goal *g* is expressed as a clause with a verb, a target, ways and qualities. For a goal, there must exist a verb $v \in \mathrm{VE}$ (a verb set) and a target $t \in \mathrm{TA}$ (a target set) expressed in domain ontology. The target designates an entity affected by the goal. The verb and target of a goal *g* can be obtained by the two functions *verb* and *target*, respectively. Another component of a goal is way, which is represented by a set of parameters whose values specify the means to satisfy a goal. More specifically, quality, which is represented by a set of indices, defines a mechanism to be used to evaluate the degree of satisfaction for the goal.

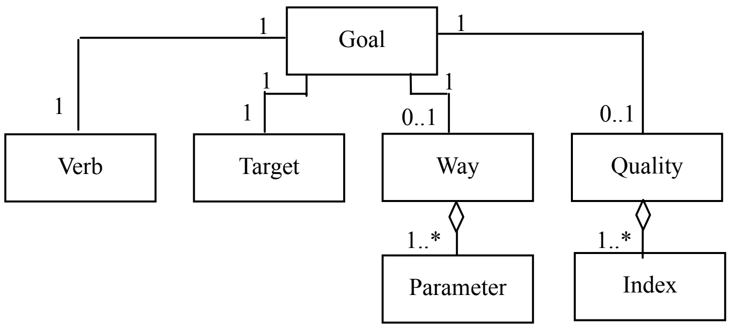


Fig. 4. The Goal Structure

For each parameter p_i of a goal with $verb(g)=v$ and $target(g)=t$, $pv(p_i, g)$ is a function to obtain the value domain for p_i of *g* and $p_i\ (g)$ is a function to obtain the value of the parameter p_i of goal *g*. A parameter set can be defined for a verb *v* and any goal whose verb is *v* will inherit these parameters. For example, a parameter called *ByCompany* is defined for verb *Inquiry,* and then goal *Inquiry* (*Ticket*) will inherit the parameter *ByCompany*.

The relationships among goals can be categorized in vertical and horizontal dimensions. In the vertical dimension, the relationships defined between the high-level goals and the lower level goals include specialization and decomposition.

1) Specialization Relationship

If realizing another goal g_2 can satisfy a goal g_1, then we call g_2 specialize g_1 and denote it as $g_2 \in SP_g(g_1)$. Accordingly, we can call g_1 generalize g_2 and denote it as $g_1 \in GE_g(g_2)$.

If $verb\ (g_1)$ is different from $verb\ (g_2)$ then their specialization relationship should be defined explicitly. If $verb\ (g_1)= verb(g_2)$, then specialization relationship can be determined using the following reasoning mechanisms:

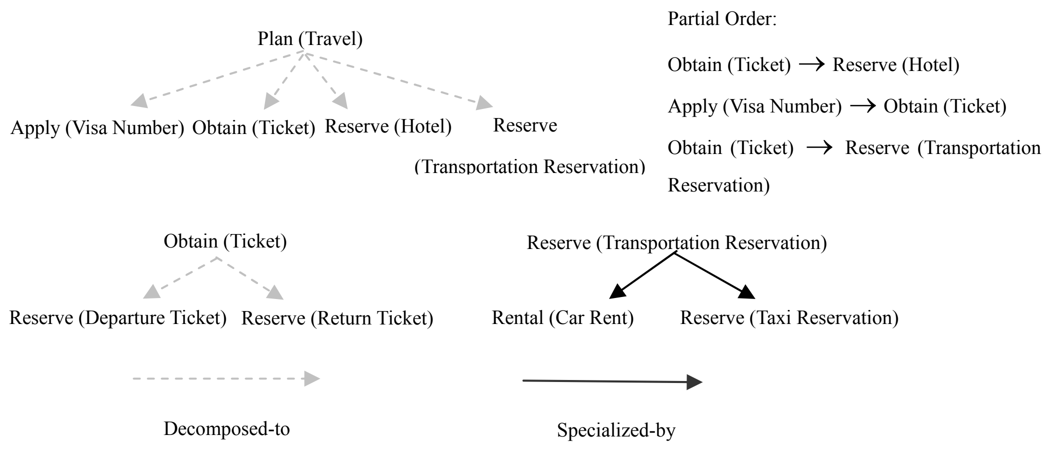

Plan (Travel)
Apply (Visa Number)
Obtain (Ticket)
Reserve (Hotel)
Reserve (Transportation Reservation)
Partial Order:
Obtain (Ticket) → Reserve (Hotel)
Apply (Visa Number) → Obtain (Ticket)
Obtain (Ticket) → Reserve (Transportation Reservation)
Obtain (Ticket)
Reserve (Departure Ticket)
Reserve (Return Ticket)
Reserve (Transportation Reservation)
Rental (Car Rent)
Reserve (Taxi Reservation)
Decomposed-to
Specialized-by

Fig. 5. Goal Relationships for Travel Plan

Given two sets A and B, for $\forall a \in A$, we have $a \in B$, or $\exists b \in B$ and $a \in SP_t(b)$, then $A \subseteq SP_s(B)$ (A specializes B).

Suppose a goal g, where $verb(g)=v$ and $target(g)=t$ with parameter set $P=(p_1, p_2,..., p_n)$ and a goal g', where $verb(g')=v$ and $target(g)=t'$ with parameter set $P'=(p_1, p_2,..., p_n,\dots, p_m)$. If $t' \in SP_t(t)$ or $t'=t$, and for $\forall pv(p_i, g'), pv(p_i, g') \subseteq SP_s(pv(p_i, g))$, then g' specializes g. For example, *Reserve* (*Train Ticket*) $\in SP_g(Reserve(Ticket))$.

2) Decomposition Relationship

A goal g can be decomposed into several sub-goals $G'=\{g_1, g_2, \dots, g_m\}$ and each sub-goal $g_i=SubOf(g)$ (i=1, 2, …, m) will contribute to the partial fulfillment of g. In order to avoid confusion for concepts, $verb(g_i)$ must be different from $verb(g)$. For the goal g, a sub-goal g_i can be optional or indispensable.

In the horizontal dimension, there are two types of relationship that can be defined among the sub-goals of g:

1) Ordering Relationship

If g_1 and g_2 are two sub-goals, and g_1 must be fulfilled before g_2, then g_1 is said to have precedence over g_2 and is denoted as $g_1 \rightarrow g_2$.

2) Dependency Relationship:

- AND(g_1, g_2): If g_1 should be satisfied, then g_2 should also be satisfied and visa versa.
- XOR(g_1, g_2): Either g_1 or g_2 is selected to be satisfied.

Figure 5 shows the goal relationships defined for the travel plan problem.

3.3 An Event-Condition-Action Rule based Process Model and Process Component

A process can be regarded as an approach to satisfy certain goals, which are defined by a set of activities, ordering constraints and the data exchange among the activities. It can be either abstract or concrete. An abstract process represents a process pattern, which can be instantiated into a concrete process to fit for a specific context. For a high level abstract goal, different processes are selected and composed together.

The process component pc, which is used to model a reusable process unit, can be represented by $<g_p, p_m, I_p, O_p>$, where g_p is an operation goal, p_m is a process model, I_p and O_p are input objects and output objects, respectively. The functions *goal* and *process* are defined as follows:

goal: $PC \rightarrow G$

process: $PC \rightarrow P_m$

where PC is the set of process components, G is the set of goals and P_m is the set of process models. The function $goal(pc_i)$ gives the goal g_i associated with pc_i, and the function $process\ (pc_i)$ gives the process model p_{mi} associated with pc_i.

3.3.1 Event-Condition-Action Rule and Process Model

A process model can be specified using different languages, for example, XPDL (WfMC, 2000), BPEL4WS (IBM, 2003) or BPML (BPMI, 2001), with differences in their syntax and expressive power. Comparisons of these different languages can be found in some papers (van der Aalst, 2002; Peltz, 2003). This work describes a process model using event-condition-action (ECA) rules. It allows customers to incrementally and interactively change the executing process model.

An ECA rule can be defined as $R_{eca}=<e, C_{on}, A_c>$, where e is an event to trigger the rule, C_{on} is the condition set to reflect the status of the system and environment, and A_c is the action set. An ECA rule states that if e happens and the condition set C_{on} is satisfied then the rule will be fired and all the actions of A_c are executed. An ECA rule can also be denoted as:

On e If C_{on} Then A_c

Events can be composed using the following simple operators:

1) AND: e_1 AND e_2 implies that both e_1 and e_2 must happen.
2) OR: e_1 OR e_2 implies that at least one of e_1 and e_2 happen.
3) PRE: e_1 PRE e_2 implies e_1 happens before e_2.
4) REP: REP e_1 n implies e_1 happens and repeats n times.

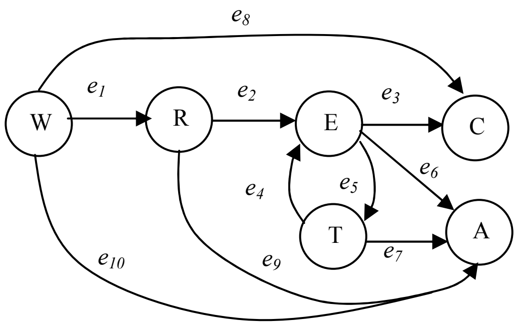


Fig. 6. The State Transition Chart of Activity

Based on these simple operators, more complex operators can be defined. For example:

ALL: ALL $(e_1, e_2, \dots, e_n)=e_1$ AND e_2 AND … AND e_n

A process model is defined as $PM=<A_t, RA>$, where A_t is the activity set of a process; RA represents relationships among the activities. $RA=DF \cup LF$, where DF and LF are respectively data flow set and the logic ordering relationships (or control flows) among the activities. DF and LF are both represented using ECA rules.

An activity can be in one of the following states: waiting (W), ready (R), executing (E), completed (C), overtime (T) and aborted (A). When an activity changes its state from one to another, an atomic event happens. Figure 6 shows the state transition chart of an activity. ECA rules can be used to represent the state transitions in an activity. The rule to transform the state of an activity from "waiting" to "ready" is called triggering rule for the activity.

The content of each activity in a process model can be specified by an operation, a process component or a requirement goal. The operation can be a predefined action (for example, data format transformation), an application invocation, a service invocation, or a task performed by human. For example, in a "*Reserve* (*Ticket*)" process model shown in Figure 7, the activity a_2 will display the query results to the customer and the customer then selects one from the results. In the case that a specific process component is defined as the content of an activity, when the state of the activity is changed to “Ready”, the process component will be instantiated as a sub-process of this activity.

The content of an activity can also be set to another requirement goal, which guides the underlying system and the customer to choose a process component during run time. For example, in Figure 7, activity a_1 defines a goal to inquire about the ticket information. If there are several activities whose contents are defined as different goals, semantic relationships among these activities are defined as constraints among targets, parameters of different goals. For example in Figure 7, the goal of a_1 is g_1: *Inquire* (*Ticket*) and the goal of a_3 is g_2: *Book* (*Ticket*). The semantic relationships between these two goals can be:

Target (g_1)=*Target* (g_2)

ByCompany (g_1)=*FromCompany* (g_2)

These two semantic relationships impose the constraints that the types of ticket and companies should be the same for these two goals. When g_1 is specialized by instantiating the target object and parameter values after the process has been started, g_2 will be specialized accordingly based on the semantic relationships.

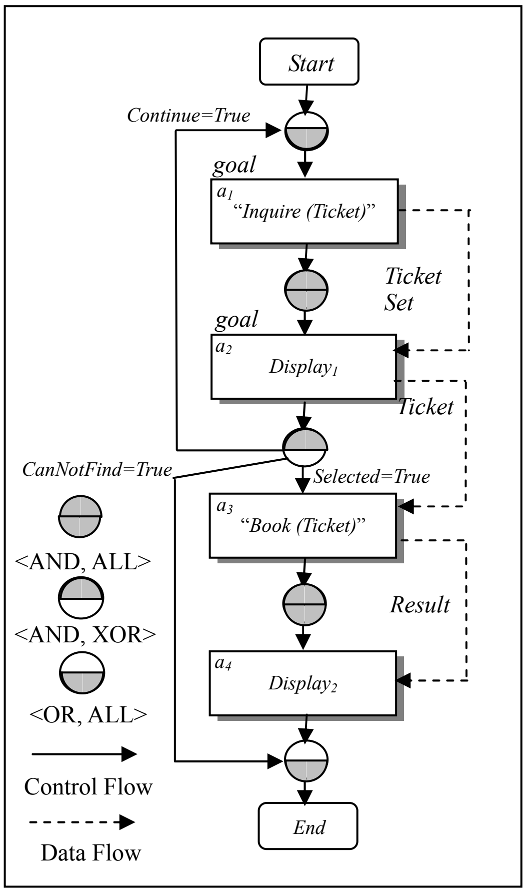


Fig.7. A Process Model for “*Reserve* (*Ticket*)”

So far the ECA rule based process model has been introduced. In the model, activities are activated by the ECA rules, and their relationships are not defined explicitly. By adding or modifying ECA rules, the process can be built and refined incrementally. ECA rule based process model also has strong expressive powers. All patterns used in defining a workflow can be expressed in terms of ECA rules.

3.3.2 The Process Graph for Process Component Modeling

Designing a new process model by explicitly writing ECA rules is quite cumbersome. Hence, the process graph is introduced to facilitate the design of process components. An example of a process graph is shown in Figure 7. The solid arrow represents control flow and the dashed line represents data flow. In the process graph, each control flow corresponds to an event. The purpose of introducing the logic nodes is for the composition of events. When the source of a control flow is an activity, the corresponding event of this flow is one of state changing events of the activity. If the source of a control flow is a logic node, then its corresponding event *e* will be determined according to the types of logic node itself.

Suppose {e_1, e_2, …, e_m} are the events corresponding to the control flows that are introduced into this logic node:

- If the type of the logic node is <AND, ALL>, then e=e_1 AND e_2 AND … AND $e_{m,}$;

- If the type of the logic node is <OR, ALL>, then $e=e_1$ OR e_2 OR…OR e_m,
- If the type of the logic node is <AND, XOR>, then the outputting control flow corresponds to the event indicating the satisfying state of the condition related to this node. An ECA rule will be added to map the satisfying state of the condition to a new event *e*:

  On e_1 AND e_2 AND … AND e_m
  If *Condition=True*
  Then *RaiseEvent*(*e*)

  Where action *RaiseEvent*(*e*) generates an atomic event *e* to indicate that the condition is satisfied.

It is easy to convert a process graph into an ECA rule set. For example, the process model shown in Figure 7 can be translated into the ECA rule set as shown in Table 1.

Table 1 ECA rules for the Process Shown in Figure 7

Activity Set: *Start*, a_1, a_2, a_3, a_4, *End*	On e_2 Then Initialize (a_3)
Control Flows:	On EndOf(a_3) Then Initialize(a_4)
On EndOf(*Start*) OR e_1 Then Initialize(a_1)	On EndOf(a_4) OR e_3 Then Initialize(*End*)
On EndOf(a_1) Then Initialize (a_2)	Data Flows:
On EndOf(a_2) If Continue=True Then RaiseEvent (e_1)	On EndOf(a_1) Then InputData(Ticket Set, a_2)
On EndOf(a_2) If Selected=True Then RaiseEvent (e_2)	On EndOf(e_2) Then InputData(Ticket, a_3)
On EndOf(a_2) If CanNotFind=True Then RaiseEvent (e_3)	On EndOf(a_3) Then InputData(Result, a_4)
EndOf( ): event indicating an activity status changing from "executing" to "completed"	
Initialize( ): Turn the state of an activity from "waiting" to "ready"	
InputData( ): Input a data object to an activity	

3.3.3 The Process Component for Service Invocation

In the service customization model, a Web service invocation is treated as a standardized process component. The general structure of this process component is depicted in Figure 8. Wrapping Web service invocation into a process component helps resolve several important problems:

(1) Semantic Description: Since it has been wrapped into a process component, the operation goal is added as its semantic description.

(2) Semantic Discrepancy: In the process component, input data is transformed into a strongly typed data that a Web service employs, and the result is transformed into a specific format of a specific system.

(3) Quality of Service: Only trusted Web services can be wrapped into process components. Exception handlings are also added to the process component to deal

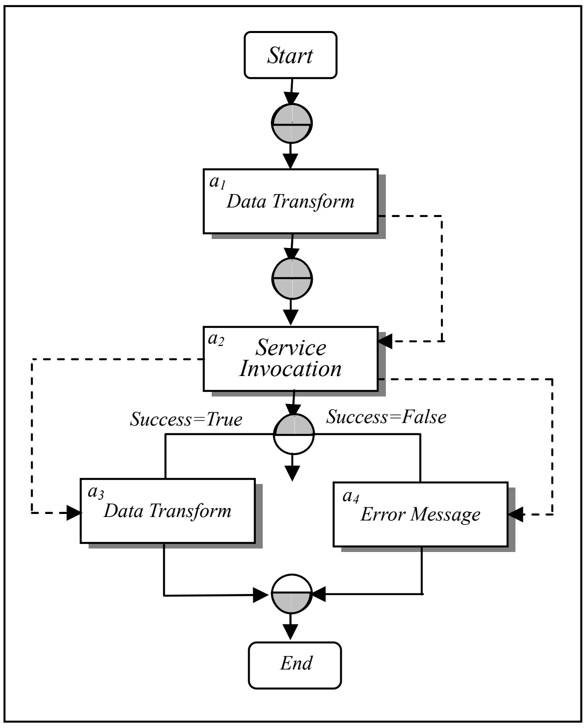


Fig.8. The Process Component for Service Invocation

with the requirements of quality of service.

With the model of customizable service process introduced, the next section describes the mechanism of how to customize a service process, i.e., the key technologies to support service customization.

4. The Service Customization Method

In order to make the best use of a set of models available within a service offering system to provide multiple functions to satisfy customers' needs, the knowledge contained within the system needs to be integrated with the implicit knowledge of customers. To achieve this goal, the common approach to passively answer customer's requests does not work. Instead, a mechanism that utilizes an interface to allow a customer to define and change the service process interactively should be provided. To better illustrate this approach, the method for implementing the customer interface is first introduced. Then the process engine and the methods supporting the choice of appropriate process components are discussed.

4.1 The Customer Interface for Service Customization

The customer interface for service customization is composed of a set of operations as shown in Figure 9. More specifically, the following operations are provided:

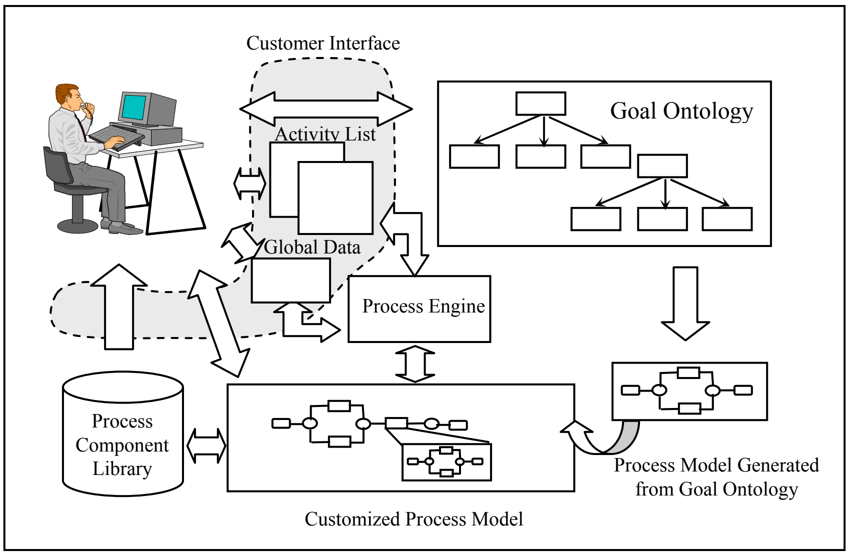


Fig. 9. The Service Customization Method

- O_1: *Sub-Goals Selection*: since a goal is decomposed into sub-goals, the sub-goals can be selected based on customers' requirements;
- O_2: *Goal Specialization*: the customer can specialize a goal to express more specific requirements;
- O_3: *Process Component Choice:* the customer can search and choose a process component for a goal interactively;
- O_4: *Data Input*: the customer can input data that is needed by processes or activities;
- O_5: *Data Mapping*: the customer can map data for activities to a global data format or visa versa;
- O_6: *Control Flow Modeling*: the customer can add control flows among activities;
- O_7: *Activity Information Access*: the customer can obtain activity information from activity

lists;

- *O_8*: *Activity Execution*: the customer can execute a concrete activity.

There are two activity lists maintained within the interface. One list, denoted by AL_w, stores activities with the state ”waiting”, and the other list denoted by AL_r, stores the activities with state “ready”. If the state of an activity is “ready”, it means that this activity can be executed once the required input data objects are available. If the state of the activity is “waiting”, it means that the activity cannot be executed because none of its triggering rules has been fired.

The method for service customization is illustrated using the travel plan example. A customer initializes a new service process by picking a goal, for example *Plan* (*Travel*). Since the goal *Plan (Travel)* is abstract and needs to be decomposed into sub-goals, the customer makes a choice from these sub-goals, for examples, *Obtain* (*Ticket*), *Reserve* (*Hotel*) and *Reserve* (*Transportation Reservation*). The selected goals should not have any conflicts with dependency relationships defined. After the customer selects the sub-goals, as denoted here by the set *G*= {*Obtain* (*Ticket*), *Reserve* (*Hotel*), *Reserve* (*Transportation Reservation*)}. The partial orders among these sub-goals can also be obtained, as denoted by the set *P*={*Obtain* (*Ticket*)-> *Reserve* (*Transportation Reservation*), *Reserve* (*Ticket*)-> *Reserve* (*Hotel*)}>. These sub-goals and their ordering relationships are transformed into an ECA rule-based process model. Firstly, each goal $g_i \in G$ is mapped into an activity a_i. Then for each ordering relationship $g_i \rightarrow g_j$ in *P*, an ECA rule can be generated as follows:

R: On EndOf(a_i) Then Initialize(a_j).

If another ordering relationship $g_k \rightarrow g_j$ exists, then *R* is changed into:

R: On EndOf(a_i) AND EndOf(a_k) Then Initialize(a_j)

In short <*G*, *P*> is transformed into a process model <*A*, *LF*>, where for the travel plan example, we have:

A={a_1: *Obtain (Ticket)*, a_2: *Reserve* (*Hotel*), a_3: *Reserve* (*Transportation Reservation*)}
LF={On EndOf(a_1) Then Initialize(a_2),
On EndOf(a_1) Then Initialize(a_3)}

The customer can add control flows to the process model interactively. Since the process model is based on ECA rules, a new ordering relationship can easily be added for the activities: after selecting an activity from AL_w, the customer chooses one event from the event list as the triggering event for the activity or edit the event expression of the triggering rules for the activity. For example, if the customer wants to define an ordering relationship, reserving transportation after making hotel reservation, the triggering rule of a_3 can be changed to:

On EndOf(a_1) AND EndOf(a_2) Then Initialize(a_3)

In the activity set *A*, some activities, for example a_2 and a_3, will be triggered by ECA rules defined, while others cannot be triggered because the knowledge contained within the process framework may not be enough to automatically trigger them. The customer should start those activities, for example a_1. These activities are appended into AL_r and the activities that will be triggered using ECA rules are added into AL_w. For the example, a_1 is added into AL_r and the other two activities a_2 and a_3 are added into AL_w.

The customer selects an activity from AL_r, say a_1, which becomes a goal according to it’s content. Since this goal is decomposed into two sub-goals in the goal ontology, i.e., *Reserve* (*Departure Ticket*) and *Reserve* (*Return Ticket*) respectively. A sub-process model P_1: <A_1, LF_1> for activity a_1 is built following the steps described earlier. Since none of the ordering relationships are defined between these two goals, LF_1 is null. For $\forall a_{1i} \in A_1$, a_1 becomes a_{1i}'s parent, a_{1i} is a_1's child and this relationship is denoted as $a_1 = Parent\ (a_{1i})$. a_1 is also called a complex activity. The service process is now structured hierarchically. The activities of A_1 will be added to AL_r or AL_w; in the example, they will be added to AL_r.

If the content of an activity selected from AL_r is a goal and this goal can't be decomposed further according to the goal ontology, then a process component should be selected for it, for example, activity a_{11}: *Reserve (Departure Ticket)*. The details for how to select an appropriate process component will be discussed at section 4.3. For now, suppose the selected process component is pc_1 as depicted in Figure 7, then the content of a_{11} will be changed to pc_1.

In order to execute the process instance instantiated from pc_1, a local data object set is built for this instance. The customer should set up the mapping relationships between the local data objects and the global data objects. According to the domain ontology, for each local data object, the customer only needs to select one from a list of global data objects with the same type to set the mapping relationships.

During the mapping process for the input data objects, the customer can encounter one of following three cases:

- All input data is available;
- Some input data is missing, but the data can be provided by the customer;
- The customer cannot provide the missing data without carrying out other activities.

The third case suggests that another process needs to be executed because of the lack of enough available information resources for the current process and goal. For example, the customer, who needs to book a foreign flight ticket, may also need a visa. The system should find a process component that helps the customer apply for the visa. The component is easy to find since the data has already been defined within each process component. Suppose the process component is pc_2, an activity (suppose it is denoted as a_4) whose content is pc_2 will then be added into AL_r. Since activity a_{11} has to wait for the data from a_4, then a_{11} can be moved into AL_w and its state is changed to "waiting". A triggering rule is added:

On EndOf(a_4) Then Initialize(a_{11})

For each activity in AL_w, if its goal has ordering relationships with the newly added goal *goal*(pc_2) according to the goal ontology, the triggering rules should also be updated.

4.2 The Process Engine for Event-Condition-Action based Process Model

As shown in Section 4.1, instead of defining a process model in advance, the customer dynamically refines the executing process model interactively. As the process is executing, customer no longer deals with the whole process model, but works primarily with the sub-models, which consist of the activities that are in the activity lists and their relationships. The process engine should continue to update the process model and eliminate the activities that have been executed. Updating the process model can improve the customizability of the service process and also reduces the complexity of process customization and composition tasks.

This process model can be updated by rewriting the ECA rules. The rule rewriting method is based on the idea that an event expression can be simplified (reduced) as the events have been executed and thus can be removed from the expression.

For example, for an ECA rule:

On EndOf(a_1) AND EndOf(a_2) Then Initialize(a_3)

If a_1 has been executed, then the value of event EndOf(a_1) can be set to *True*. The rule can now be simplified as:

On EndOf(a_2) Then Initialize(a_3)

The rules for rewriting event expressions are shown in Table 2. In the table, E denotes a composite event and e_1 is an event included in the event expression of E.

Table 2 Reduction Rules for Event Expression

1. E/e_1=E，if $e_1 \notin$E	4. (e_1 PRE e_2)/ e_1= e_2
2. (e_1 AND e_2)/ e_1= e_2	5. (e_2 PRE e_1)/ e_1=False
3. (e_1 OR e_2)/ e_1=*True*	6. REP e_1 n/ e_1=REP e_1(n-1)

As an example, for e=(e_1 AND e_2)PRE（e_3 OR e_4）, if e_1，e_2，e_3 and e_4 have happened in a successive order, then:

After e_1 happened, e=e_2 PRE (e_3 OR e_4)

After e_2 happened, e=e_3 OR e_4

After e_3 happened, e becomes True

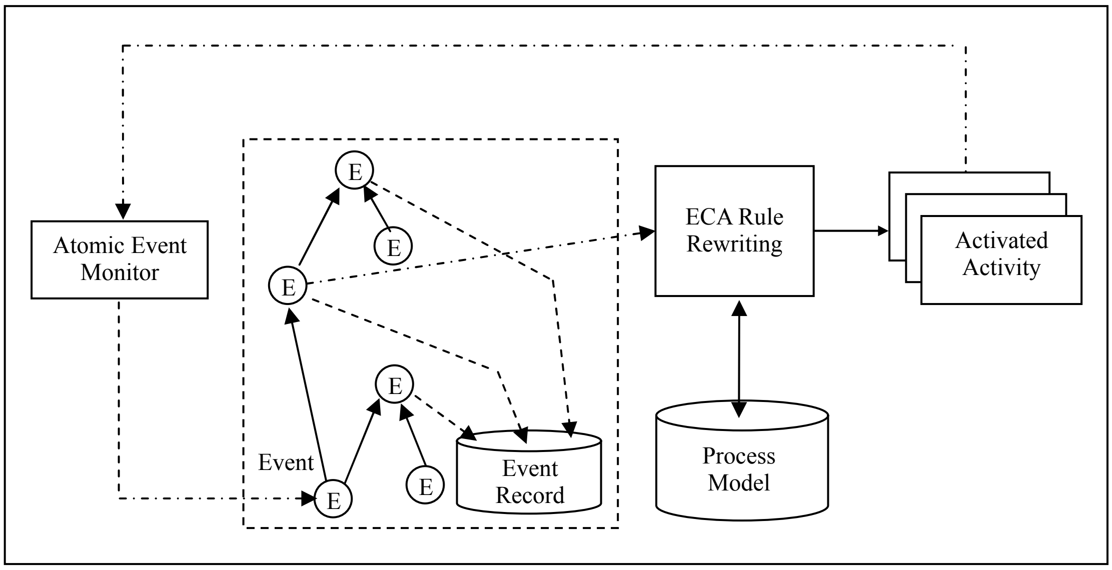


Fig.10. The Executing Mechanism of the Process Engine

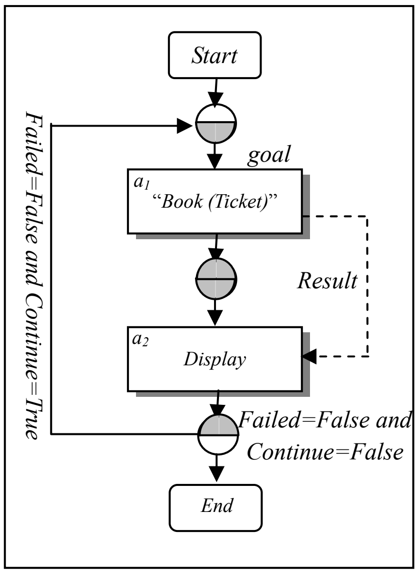
Figure 10 shows how the process engine works. When an atomic event occurs, the process engine will detect and record the event with the atomic event monitor. The event and its results for related composite events will also be detected and recorded. ECA rules are rewritten accordingly to update the process model. When an ECA rule is triggered, the related activity will have it's state changed into "ready" and be activated. As an activated activity is executed, a new event occurs.

It can be observed that the proposed engine executes a process based on an event-triggering mode so that it still works when the process model is not completely defined.

4.3 Process Component Search Strategy

When customizing a process, another key issue is the search strategy for the process components, which are the basic building blocks for the process composition. For a goal that cannot be further divided into sub-goals, the tasks of its process components are executed to satisfy the goal. For a goal denoted by g, all the process components whose operation goals specialize *g* should be found. If no eligible process components are found, set *target* (*g*) $\in GE_t$ (*target* (*g*)) so that *g* is replaced by *g*' with the assumption that a process component for the generalized object will also be fit for the specialized object. For *g*', a new search is then started for *g*'. This procedure continues until a set of eligible process components are found.

For the travel plan example, suppose there are not any process components that are found to satisfy the goal *Reserve*(*Departure Ticket*), then the search is to find the process components whose operation goals specialize the goal *Reserve* (*Ticket*) since *Ticket*$\in GE_t$ (*Departure Ticket*). Figure 7 already shows one candidate process component with the operational goal *Reserve (Ticket) By("Inquire First")*, which allows the customer to input query conditions and to select one ticket from the results. The process can iterate repeatedly until satisfying result is obtained. Figure 11 shows an alternative process component with operational goal *Reserve* (*Ticket*) *By*("*Book Directly*"), the customer can book the train or flight directly with known information. Since there are only two process components for choice so that it is quite easy to make a decision. If the customer does not know any information, the process component shown in Figure 7 will be selected.


Start
goal
a_1
"Book (Ticket)"
Result
a_2
Display
Failed=False and Continue=False
End
Failed=False and Continue=True
Fig.11. Another Process Component for "*Reserve (Ticket)*"

1) A Decision Tree Model for Process Component Choice

It is difficult for the customer to make a choice when many process components are resulted from a search. For example, in the process model shown in Figure 7, the task of the first activity is to select a process component for it's goal *Inquire* (*Ticket*). For this goal, a set of process components that can specialize it are found but they have different values of targets and parameters defined in the *way* of the goal structure. Suppose the following operation goals of process components are found:

- *Inquire (Train Ticket) ByCompany* (*Company A*)
- *Inquire (Train Ticket) ByCompany* (*Company B*)
- *Inquire (Flight Ticket) ByLocation (Airport) ByCompany* (*Company C*)
- *Inquire (Flight Ticket) ByLocation (City) ByCompany* (*Company C*)
- *Inquire (Flight Ticket) ByLocation(City) ByCompany* (*Company D*)

The customer now is facing to make a decision to select from these choices. A decision tree based model can be employed to help making selection among the process components. Figure 12 shows an example of a decision tree model for selecting a process component for the goal *Inquire* (*Ticket*). In this decision tree, the decision choices are classified by targets and common

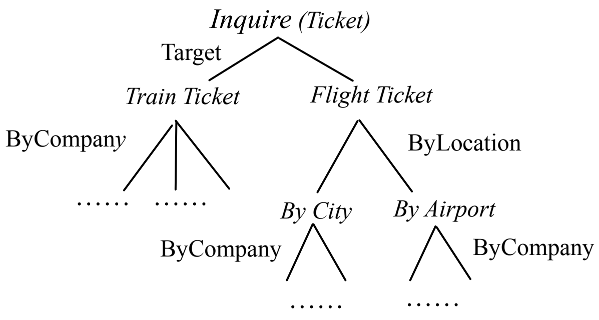


Fig.12. A Decision Tree Model Example

parameters defined in the way of the goal structure.

2) A Quality Measurement Model for Decision Making

Another issue in the decision-making is how to make a proper decision when traversing through a decision tree. For example, which type of ticket should the customer reserve, train ticket or flight ticket? How to select a company from hundreds of travel agencies? It is difficult to develop a pre-defined procedure within a travel planning system that can make an optimized decision for each customer. The issues is to what degree can decision supports be provided for a particular customer?

A quality measurement model is proposed to help the customer to make decisions. For a goal g, the target can have different values and the parameters of the way can also have different values. Utility functions can be defined for target and parameters. The utility function for a parameter can be defined as cf_i: IS× $pv(p_i, g)\rightarrow[0,100]$, where IS is the index set of g, $pv(p_i, g)$ stands for the value domain of parameter p_i of g. The definition of the utility function for target is similar and it is denoted as cf_0. The utility value represents the quality of each index.

Given a goal, the quality values for a target and a parameter can be measured respectively by:

$$QoG_0 = \sum_{i=1}^{n} \left(\frac{w_i}{\sum_{k=1}^{n} w_k} \cdot cf_0(I_i, target(g)) \right)$$

and

$$QoG_j = \sum_{i=1}^{n} \left(\frac{w_i}{\sum_{k=1}^{n} w_k} \cdot cf_j(I_i, p_j(g)) \right)$$

where n represents the total number of indexes for the target or a specific parameter and $w_i \in [1,10]$ is the weight given to the index I_i according to the customers' preferences. For example, suppose the index set of *Inquire (Ticket)* is {Time, Cost, Comfort, Easy to Operate}. Now the customer is making a choice between different values of the target, for traveling by train or flight. Assume the utility values are given as follows:

For g_1=*Inquire* (*Train Ticket*):

cf_0 (Time, Train Ticket)=30, cf_0 (Cost, Train Ticket)=90, cf_0 (Comfort, Train Ticket)= 40, cf_0 (Easy to Operate, Train Ticket)=80

For g_2=*Inquire* (*Flight Ticket*):

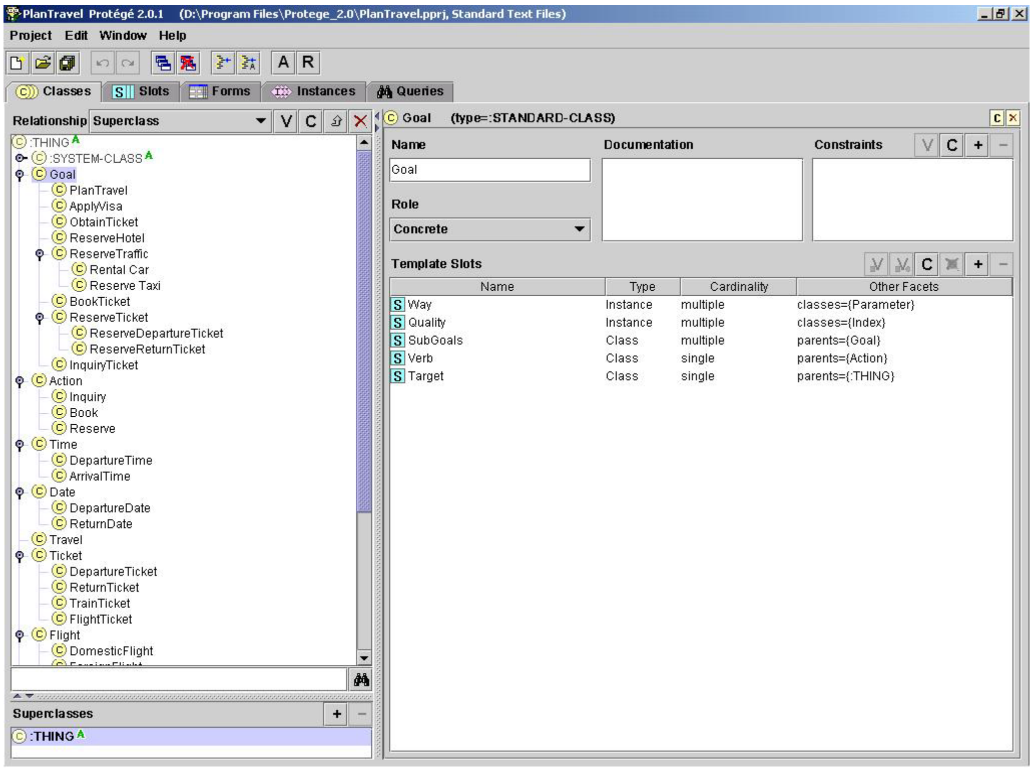
cf_0 (Time, Flight Ticket)=90, cf_0 (Cost, Flight Ticket)=20, cf_0 (Comfort, Flight Ticket)=70, cf_0 (Easy to Operate, Flight Ticket)=50

Furthermore, let the weights assigned to indexes are 9, 2, 6 and 1 respectively. The quality values are thus obtained as *QoG*=42.7 for g_1 and *QoG*=73.3 for g_2. The system suggests the customer to choose *Inquire (Flight Ticket)*.

As we pointed out in Section 2, the knowledge contained within the system is not always sufficient for satisfying all the customers' decision support requirements. Therefore, the results obtained from the quality measurement can only serve as a reference for the customer. In practice, it may be possible to add more indices and define more complex utility functions based on the history of service offerings and other information. For example, a utility function man be used to calculate the index based on the travel time needed. Although such utility function may provide more accurate measurement for the quality, it may also add more complexity to the system.

5. Service Customization Center (SCC): A Prototype System for Supporting Service Customization

We have developed a prototype system that includes several tools and a hypothetical Service Customization Center (SCC) to support service customization based on the methodology discussed in this paper. Protégé is used for domain and goal ontology modeling. It is an open-source, Java tool that provides an extensible architecture for the creation of customized knowledge-based applications (SMI, 2002). In the view of Protégé, an ontology is a formal explicit description of concepts in a domain of discourse (classes, sometimes called concepts), properties of each concept describing various features and attributes of the concept (slots, sometimes called roles or properties), and restrictions on slots (facets, sometimes called role restrictions). In the prototype system, we define the domain ontology and goal ontology in Protégé. The ontology together with a set of individual instances of classes constitute the knowledge base of SCC. SCC accesses domain ontology and goal ontology through the APIs provided by Protégé. Figure 13 shows the definition of domain ontology and goal ontology for travel planning in Protégé.



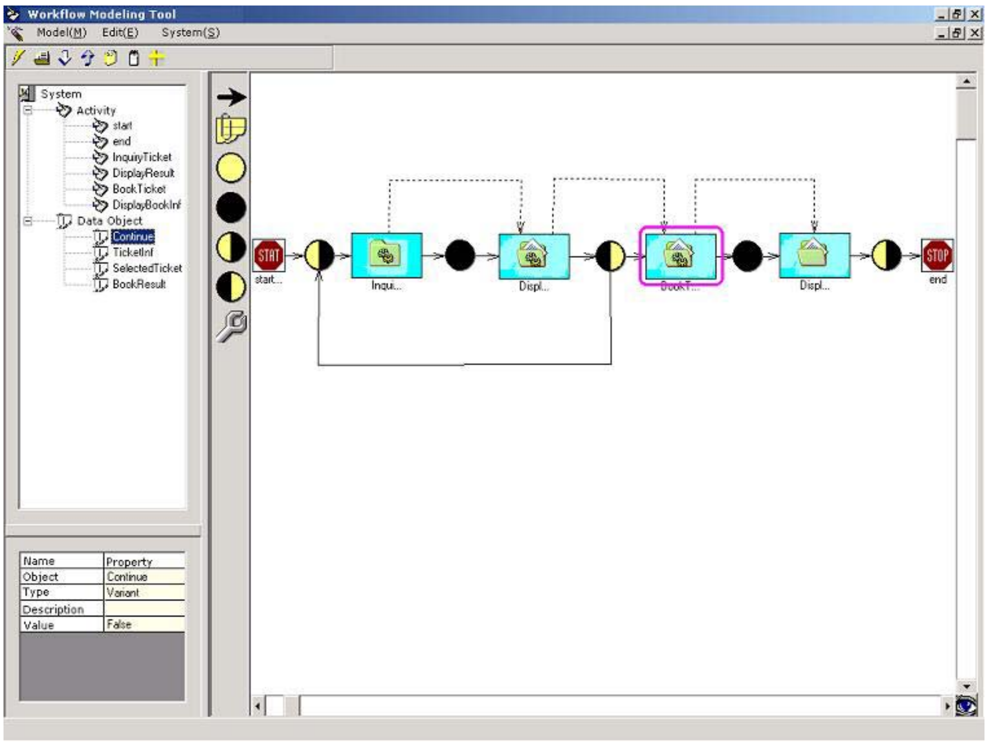
Name: Goal

Role: Concrete

Name	Type	Cardinality	Other Facets
Way	Instance	multiple	classes={Parameter}
Quality	Instance	multiple	classes={Index}
SubGoals	Class	multiple	parents={Goal}
Verb	Class	single	parents={Action}
Target	Class	single	parents={:THING}

Fig.13. Ontology Modeling using Protege

Figure 14 shows a process component-modeling tool for the prototype system. This tool is flexible and easy to use. It supports the drag and drop operations to draw the process graph. It is composed of a navigation tree to show the hierarchical structure of all the entities in the process (left panel), and a composition panel for the ECA rule based workflows (right panel) as shown in Figure 14. To assist the user to compose a complex process model, a validation mechanism is supported. The validation mechanism can verify the input information on the spot. A special event algorithm is implemented to deal with the validation of model when transforming the graphical representation into ECA rules. The process component designed by the tool is stored in a database

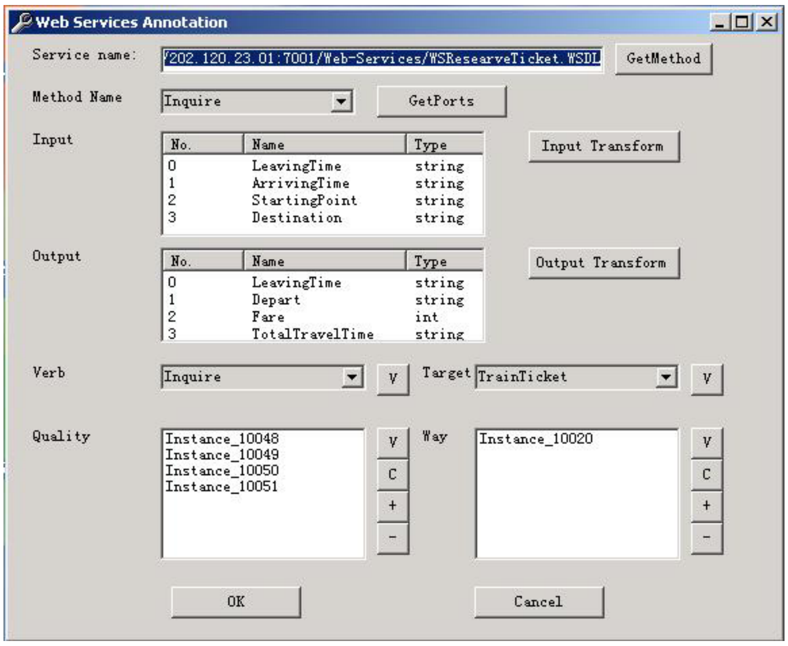


Name	Property
Object	Continue
Type	Variant
Description	
Value	False

Fig.14. Process Component Modeling

and it can be loaded from the database into the current running process. The process model shown in Figure 14 represents "*Reserve* (*Ticket*)" process that is introduced in Section 3.3.1.

If currently existing service solutions cannot support the required service customization, a new service must be generated and wrapped into a process component in our model, as described in Section 3.3.3. In order to fulfill this task in the prototype system, a tool was developed. When a new service needs to be attached, its address of the WSDL in UDDI is entered so that its input and output of the method can be explicitly obtained through the UDDI. Specifically, the parameter



Service name: /202.120.23.01:7001/Web-Services/WSResearveTicket.WSDL

Method Name: Inquire

Input

No.	Name	Type
0	LeavingTime	string
1	ArrivingTime	string
2	StartingPoint	string
3	Destination	string

Output

No.	Name	Type
0	LeavingTime	string
1	Depart	string
2	Fare	int
3	TotalTravelTime	string

Verb: Inquire V Target: TrainTicket V

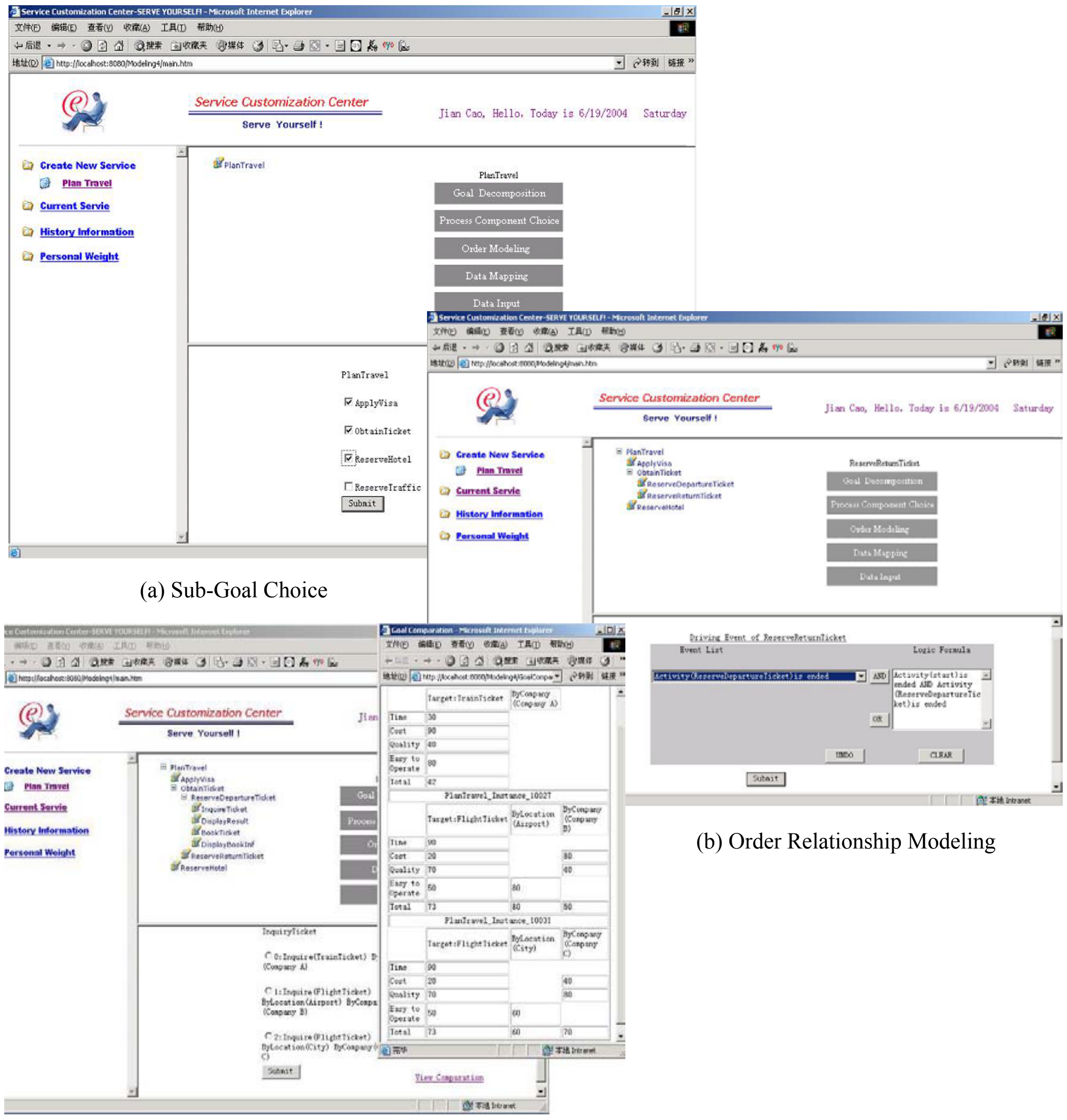
Quality: Instance_10048 Instance_10049 Instance_10050 Instance_10051 V C + −

Way: Instance_10020 V C + −

Fig.15. A Tool for Defining Web Services as Process Component

types of input and output can be obtained through parsing the WSDL file; then the transforming rules among these parameters and domain ontology are defined in XSLT format; lastly the goal description and QoS description are added so that the new service transformed into a basic process component in the system. Figure 15 shows the procedure for utilizing this tool to add a new service for inquiring train ticket information.

SCC is Web-style interface for the customer and is developed using JAVA JSP language and deployed with Jakarta Tomcat 5.0. Figure 16 (a) shows an interface for the customer to select multiple sub-goals of a travel plan. In this case, the customer chooses applying Visa, obtaining ticket and reserving hotel. These sub-goals and their relationships are transformed into a top-level service process model. Figure 16 (b) shows that the customer is defining ordering relationships among the activities. The customer can select events from an event list and compose them into an event expression of the ECA rules for the selected activity. In this case, the customer adds an ECA rule that defines the process of reserving a departure ticket ahead of reserving a return ticket.



Activity(ReserveDepartureTicket)is ended

Activity(start)is ended AND Activity(ReserveDepartureTicket)is ended

	Target:TrainTicket	ByCompany (Company A)	
Time	30		
Cost	90		
Quality	40		
Easy to Operate	80		
Total	47		
PlanTravel_Instance_10027			
	Target:FlightTicket	ByLocation (Airport)	ByCompany (Company B)
Time	90		
Cost	20		80
Quality	70		40
Easy to Operate	50	80	
Total	73	80	50
PlanTravel_Instance_10031			
	Target:FlightTicket	ByLocation (City)	ByCompany (Company C)
Time	90		
Cost	20		40
Quality	70		80
Easy to Operate	50	60	
Total	73	60	70

(a) Sub-Goal Choice

(b) Order Relationship Modeling

(c) Process Component Choice

Fig.16. A Demo Service Customization Center

Figure 16(c) shows a process component choice interface for the customer. The scores are calculated based on the weights assigned to the indices by the customer, utility values of the targets and the parameters. Figure 16(c) lists utility values of three process components for goal *Inquire* (*Ticket*). For two different targets, i.e., train ticket and flight ticket, the utility values are 42 and 73 respectively. Thus the customer will prefer to inquire flight ticket. Furthermore, the utility values for "*ByLocation*" of the two process components for inquiring flight ticket are 80 and 60, respectively. Therefore the second process component can be chosen.

We also tested the prototype system in a more complex bus manufacturing case. For a bus-manufacturing firm, delivering customizable product is very important for maintaining its competitive advantage in today's market. When a customer order is accepted, the enterprise will compose different services that are provided by different partners, such as engine vendors, part manufacturers and the shipping company. The domain ontology and goal ontology are much more complex than travel planning problem.

Although our system provides interactive service customization functionalities, which are not supported by other systems yet, we find that we need to implement extra components to build this system for such complex applications. For example, for a prototype system for supporting user defined on-demand bus ordering, we have to implement an input ontology and to define web services as process components for the system. Since ontology only needs to be entered once, it is not a critical problem and it is possible to obtain them from some public ontology library, such as the web site at http://www.daml.org/ontologies/. Currently, defining a Web service as a process component is conducted manually. It is a daunting task for a system maintainer to maintain the system when there are a large amount of Web services that need to be defined as components. It is possible to search Web services and translate them to process components automatically or semi-automatically in some limited areas.

6. Related Works

The most popular standards for building processes using Web service composition are BPEL4WS (IBM, 2003), BPML (BPMI, 2001) and DAML-S (Ankolenkar, 2002). BPEL4WS and BPML aim at abstracting the service references in the process from actual service implementations. This helps in selecting a correct service implementation for each activity during process deployment (deployment-time binding) or execution (execution-time binding). However, the present process composition standards like BPEL and BPML are inadequate for semantically representing the activity components of a process. The DAML-based Web Service Ontology (DAML-S) (Ankolenkar, 2002) is an initiative to provide an ontology markup language expressive enough to semantically represent capabilities and properties of Web services. DAML-S is based on DAML+OIL and the aim is to discover, invoke, compose, and monitor Web services. It can be used to define ontology appropriate for declaring and describing services using a set of basic classes and properties. However, DAML-S itself is not a solution for the service customization problem.

Various investigations to provide adaptability of service processes have been initiated. For example, eFlow from HP laboratories models composite service as a graph, which defines service nodes, event nodes or decision nodes (Casati et al., 2002). The eFlow engine offers the facility of being able to plug in new service offerings and enables adaptiveness. A Web Services Modeling Framework (WSMF) has been proposed to enable flexible and scalable e-commerce using Web

Services (Fensel and Bussler, 2002). It advocates using semantic Web techniques to deal with the problems of heterogeneity and scalability in e-commerce. MWSCF (METEOR-S Web Service Composition Framework) allows user to semantically define each activity involved in a process (Sivashanmugam et al., 2003). Each activity in the process can be specified using a Web service implementation, Web service interface, or semantic activity template. When the activity is specified as a semantic activity template, the activity requirements are given as the semantics of the inputs/outputs (IO) along with the functional semantics of the activity being specified.

Planning and reasoning originated in AI research has been investigated and applied in service composition. For example, Aiello et al. presented a way to compose e-Services based on planning under uncertainty and constraint satisfaction techniques, and a request language for specifying client goals was proposed (Aiello et al., 2002). McIlraith and Son addressed the composition of e-Services by using the Situation Calculus-based programming language CONGOLOG and more specifically, component e-Services were represented as CONGOLOG programs, while the client's needs were specified through suitable forms of constraints (McIlraith and Son, 2002).

The service customization model described in this paper differs from the works above in that the model can configure not only the content of an activity, but also the whole process structure according to the customer's requirements. The model is not based on planning theory, since we believe customer's needs are often subjective and implicit, and therefore, it's very difficult to specify complete requirements in advance. The model emphasizes on integrating system knowledge and customer's knowledge.

Kim and Gil introduced a framework for interactive composition of services that assists users in sketching their requirements by analyzing the semantic description of the service (Kim and Gil, 2004). An analysis tool was developed to help users create complete and correct compositions of Web services. They believe that users may only have high-level or partial incomplete description of the desired outcome or the initial state, so it may be hard to directly apply automatic approaches that require explicit goal representations. Their basic ideas are quite similar to our model. The difference is that instead of letting the user design the process freely, we provide goal ontology, process components and decision models to guide the user through the service process creation. Furthermore, the process model is based on ECA rules so that it can support executing partially defined process more efficiently.

This work is also closely related to the research of adaptive workflow or dynamic workflow. To enhance the expressive power of workflow model, providing operations to revise the model dynamically (Heimann et al., 1996; Reichert and Dadam, 1998) and designing exception handling mechanism (Chiu et al., 1998) are two main methods for improving the adaptability of workflow. We find most systems can't provide different process models for different customers needs as our model does.

Some researchers adopt planning theory to generate customized workflow models. For example, Chun et al. introduced a system that can achieve customized generation of workflows by specifying governmental regulations (Chun et al., 2002). PLMflow is a dynamic workflow system that is capable of supporting non-deterministic processes such as those found in collaborative product design scenarios (Liangzhao et al., 2002). Its workflow is constructed based on business rule inferences. The advantage and disadvantage of this method are considered the same as those using planning to compose service.

Chung et al. investigated the use of ontology, agents and knowledge based planning techniques to provide support for adaptive workflow or flexible workflow management, especially in the area of new product development within the chemical industries (Chung et al., 2003). They provided a plan library to support process reuse, which was similar to our process component library. In our model, besides providing process component library to support process reuse, goal ontology and a decision support model are provided to facilitate reuse of components. We also allow customers to revise the process interactively and incrementally.

7. Conclusions and Future Works

Mass customization refers to the ability of providing customized products or services through flexible processes and at reasonably low cost with high quality. In light of new capabilities brought by Web service, customers will soon be able to directly input and interact with service providers. The providers will have to respond to a variety of requirements from customers with certain constraints from business or relevant perspectives. In this paper, a model is proposed to deal with the challenges of service customization. In the model, system knowledge and customer's knowledge are integrated. Customized service process can be created interactively based on domain ontology, goal ontology and a process component library.

So far we have developed some tools to support this model. The future works include but are not be limited to:

(1) to design more rational structure to represent complex way for a goal. Current model represents the way in a goal as a set of unrelated parameters and is not sufficient for complex problems.

(2) to investigate how to acquire knowledge from customer behaviors. For example, in our current model, the decision tree model is predefined for all customers. We believe specific decision tree can be built dynamically by mining customer behaviors.

(3) to develop more tools to fully support the model proposed in the paper.

Acknowledgements:

This research is supported by grand project (No.03dz15027) of the Science and Technology Commission of Shanghai Municipality. This work is also partly supported by "SEC E-Institute: Shanghai High Institutions Grid", Chinese high technology development plan (No.2004AA104340), Chinese Semantic Grid project and Chinese NSF project (No.60473092). This work was partially conducted while the first author was visiting Stanford University.

References:

Aiello M., Papazoglou, M., Yang, J., 2002, A Request Language for Web-Services Based on Planning and Constraint Satisfaction, Lecture Notes in Computer Sciences, 2444, 76-85

Ankolenkar, A., Burstein M., 2002, DAML-S: Web Service Description for the Semantic Web, Lecture Notes in Computer Science, 2342, 348-363

BPMI, 2001, Business Process Modeling Language, Available from http://www.bpmi.org/specifications.esp, 2001.8

Casati, F., Ilnicki, S., Jin, L.J., 2002, Adaptive and Dynamic Service Composition in eFlow, Technical Report, HPL-200039, Software Technology Laboratory, Palo Alto, USA, Available from http://www.uddi.org/pubs/wsdlbestpractices.pdf, 2002.3

Chiu, D., Li, Q., Karlapalem, K., 1998, Exception Handling with Workflow Evolution in ADOME-WFMS: A

Taxonomy and Resolution Techniques, CSCW'98 Workshop on Adaptive Workflow Systems , Seattle, Washington, Available from http://ccs.mit.edu/klein/cscw98/paper16, 1998.11

Chun, S., Atluri, V., Adam, N., 2002, Dynamic Composition of Workflows for Customized eGovernment Service Delivery, Proceedings of The Second National Conference on Digital Government, LA, CA, Available from http://www.digitalgovernment.org/library/library/pdf/chun.pdf, 2002.5

Chung, P.W.H., Cheunga, L., Stader, J., 2003, Knowledge-based Process Management—an Approach to Handling Adaptive Workflow, Knowledge-Based Systems, 16 (3), 149–160

Duray, R, Ward, P.T., Milligan, G.W., Berry, W.L., 2000, Approaches to Mass Customization: Configurations and Empirical Validation, Journal of Operations Management, 18, 605-625

Fensel, D., Bussler, C., 2002, The Web Service Modeling Framework WSMF, Electronic Commerce Research and Applications, 1(2), 113-137

Giacomo, P., Giuliano D. V., Leonid M., 2001, Dynamic Service Aggregation in Electronic Marketplaces, Computer Networks: The International Journal of Computer and Telecommunications Networking, 37(2), 95-109

Gruber, T. R., 1993, A Translation Approach to Portable Ontology, Knowledge Acquisition, 5(2), 199-220

Guarino, N.. 1998, Formal Ontology and Information Systems. Proceedings of Formal Ontology and Information Systems, Trento, Italy, IOS Press, Amsterdam, 1998, pp.3-15.

Heimann, P., Joeris, G., Krapp, C.A., Westfechtel, B., 1996, DYNAMITE: Dynamic Task Nets for Software Process Management, Proc. 18th Int. Conf. Software Engineering, Berlin , Germany, pp.331-341, 1996.3

IBM, 2003, Business Process Execution Language for Web Services, Version 1.1, ftp://www6.software.ibm.com/ software / developer /library/ws-bpel11.pdf, 2003.5

Liangzhao, Z., David, F., Henry, C., JunJang, J., 2002, PLMflow – Dynamic Business Process Composition and Execution by Rule Inference, Lecture Notes in Computer Science, 2444, 141-15

Kim, J., Gil, Y., 2004, Towards Interactive Composition of Semantic Web Services, 2004 AAAI Spring Symposium, Technical Report SS-04-06, Available from http:// www.isi.edu /ikcap / scec/papers/ AAAI-Symp-04-Kim-Gil. pdf, 2004.6

Malone, T.W., Crowston, K., Herman, G.A., 2003, Organizing Business Knowledge: The MIT Process Handbook, Edited by Cambridge, MA: MIT Press

McIlraith, S., Son, T., 2002, Adapting Golog for Composition of Semantic Web Services, Proceeding of the International Conference on the Principles of Knowledge Representation and Reasoning (KRR'02), 2002.4, pp 482-496.

OASIS, 2002, UDDI Version 2.04 API, Available from http://uddi.org/pubs/ ProgrammersAPI-V2.04 -Published-20020719.pdf, 2002.7

Orriëns, B., Yang, J., Papazoglou, M. P., 2003, A Framework for Business Rule Driven Service Composition, Lecture Notes in Computer Science, 2819, 14-27

Peltz C., 2003, Web Service Orchestration: a Review of Emerging Technologies, Tools and Standards, Available from http://devresource.hp.com/drc/ technical_white_papers/ WSOrch/ WSOrchestration.pdf, 2003.1

Peters, L. D., Saidin, H., 2000, IT and the Mass Customization of Services: the Challenge of Implementation, International Journal of Information Management, 20(2), 103-119

Reichert, M., Dadam, P., 1998, ADEPTflex-Supporting Dynamic Changes of Workflows Without Losing Control, Journal of Intelligent Information Systems, 10(2), 93-129

Rolland, C., Ben Achour, C., 1998, Guiding the Construction of Textual Use Case Specifications*,* Data & Knowledge Engineering Journal, 25(1-2), 125-160

Rolland, C., Souveyet, C., Ben Achour, C., 1998, Guiding Goal Modeling Using Scenarios, IEEE Transaction on

Software Engineering, 24(12), 1055-1071

Sivashanmugam, K., Miller, J., Sheth, A., Verma, K., 2003, Framework for Semantic Web Process Composition, Technical Report 03-008, LSDIS Lab, Dept of Computer Science, UGA. Available from http://lsdis.cs.uga.edu/lib/download/TR03-008.pdf, 2003.6

Srivastava, B., Koehler, J., 2003, Web Service Composition - Current Solutions and Open Problems, Proceedings of ICAPS'03 Workshop on Planning for Web Services, 2003. 6, Trento, Italy, pp28-35

SMI, 2002, SMI Report, The Evolution of Protégé: An Environment for Knowledge-Based Systems Development, SMI-2002-0943, Available from http://www.smi.stanford.edu/pubs/SMI_Reports/SMI-2002-0943.pdf, 2002.9

van der Aalst, W.M.P., Dumas M., ter Hofstede, A.H.M., Wohed, P., 2002, Pattern Based Analysis of BPML (and WSCI), QUT Technical Report, FIT-TR-2002-05, Queensland University of Technology, Brisbane, Available from http://xml.coverpages.org/Aalst-BPML.pdf , 2002.5

W3C, 2001, Web Service Description Language (WSDL) 1.1, Available from http://www.w3.org/TR/wsdl, 2001

W3C, 2003, Simple Object Access Protocol, Available from http://www.w3.org/tr/soap12

WfMC, 2000, Interoperability Wf-XML Binding, Available from http://www.wfmc.org/standards/docs/Wf-XML-1.0.pdf, 2000.8

WSUI Working Group, 2002, WSUI Executive White Paper, Available from http://www.wsui.org/doc/WSUI-wp.pdf

Biography:

Jian Cao

Dr. Jian Cao received his Ph.D degree from Nanjing University of Science & Technology in 2000. He is currently Associate Professor of the Department of Computer Science& Technology at Shanghai Jiaotong University. His main research topics include service computing, cooperative information system and software engineering. He has published more than fifty papers.

Jie Wang

Dr. Jie Wang received his Ph.D degree from Stanford University and he is currently a consulting faculty at Stanford University. Dr. Jie Wang conducts research in engineering and environmental informatics and knowledge management for sustainable development. He has also worked for a number of companies and government agencies including Collation, Inc., EDS, Loudcloud, Inc, Stanford University ITSS, Instantis, Inc., First Union Bank, Department of Energy, and NOAA.

Kincho Law

Prof. Kincho Law received his Ph.D. degree from Carnegie-Mellon University in 1981. He is currently Professor of Civil and Environment Engineering at Stanford University. Professor Law's professional and research interests focus on the application of advanced computing principles and techniques to structural and facility engineering. His work has dealt with various aspects of computational science and engineering, computer aided-design, regulatory and engineering information management, and engineering enterprise integration.

Shensheng Zhang

Prof. Shensheng Zhang received his Ph.D degree from Stanford University in 1988. He is currently Professor of the Department of Computer Science& Technology at Shanghai Jiaotong

University. His main research topics include distributed computing, agent, virtual reality and software engineering. He is the Director of Computer Integration Technology Lab at Shanghai Jiaotong University

Minglu Li
Prof. Minglu Li received his Ph.D degree from Shanghai Jiaotong University in 1996. He is currently Professor of Department of Computer Science& Technology at Shanghai Jiaotong University. His main research topics include grid computing, image processing, and e-commerce. He is the Director of IBM-SJTU Grid Research Center at Shanghai Jiaotong University.