Available at
www.ComputerScienceWeb.com
POWERED BY SCIENCE DIRECT

Computer Standards & Interfaces 25 (2003) 103–117

COMPUTER STANDARDS & INTERFACES

www.elsevier.com/locate/csi

Designing a composite e-service platform with recommendation function

Duen-Ren Liu*, Minxin Shen, Chiu-Ting Liao

Institute of Information Management, National Chiao Tung University, 1001 Ta Hsueh Road, Hsinchu 300, Taiwan

Received 9 August 2002; received in revised form 13 January 2003; accepted 15 January 2003

Abstract

Enterprises make e-services available via the Internet to drive new revenue streams or create new efficiencies. Composite e-services, which consist of various e-services provided by different e-service providers, are more valuable for customers. This work presents a novel platform capable of supporting e-service metadata, and modeling and recommending composite e-services. Composite e-services are modeled by using the activity diagram of Unified Modeling Language in which Event/Condition/Action (ECA) rules are employed to control the sequence of e-services enactment and to select e-service providers. Moreover, two-level e-service metadata is designed to extend Universal Description, Discovery, and Integration (UDDI) standard to enable semantic search and selection of e-services. Finally, data mining approach is proposed to discover the frequent predicates of e-services and the frequent orderings between e-services. Based on the mining result, the proposed platform provides a recommendation of top *N* composite e-services to customers.

Keywords: e-service; Composite e-service; Workflow system; Data mining; Recommendation

1. Introduction

The Internet became a platform for business transactions recently. Enterprises provide e-services via the Internet to generate new revenue or create new efficiencies. Increasing e-services are provided, for instance, on-line subscriptions, on-line payments, on-line travel reservations, real-time news services, etc. Customers can also search and acquire e-services easily on the Internet. Although e-services have received an increasing attention, several issues are still open. Effectively advertising e-services to customers is critical for e-services providers. However, conventional search engines cannot fulfill customers' needs to provide semantic search of e-services. Besides, individual e-service cannot accomplish a customer's goal; a complete service generally involves several basic e-services. For instance, a travel service may integrate an airline reservation, a hotel reservation, a car rental, a package delivery, and a ticket reservation. Thus, this work aims to aid customers to discover and compose desired e-services.

Several e-service platforms are proposed. For example, HP e-speak [9] and Microsoft.NET [13] are such platforms and share many concepts and features. Basic features of these platforms are registering, advertising, monitoring, and managing e-serv-

* Corresponding author. Tel.: +886-3-5712121; fax: +886-3-5723792.
E-mail address: dliu@iim.nctu.edu.tw (D.-R. Liu).

0920-5489/03/$ - see front matter © 2003 Elsevier Science B.V. All rights reserved.
doi:10.1016/S0920-5489(03)00012-6

ices. Composite e-service issues are widely discussed. As a workflow is consisted of many tasks, existing approaches, such as Casati and Shan [6,7], generally model a composite e-service as a process that contains many basic e-services. Balakrishnan [4] proposed a Service Framework Specification to compose e-services. The author indicates that a dynamic e-service interaction model should involve Service Interaction, Service Model, and Service Provisioning. Moreover, Piccinelli et al. [14] proposed a DysCo model which indicates that an ontology-based approach should describe the semantics and characteristics of a service. Sahai et al. [17] extends Application Response Measurement (ARM) to perform end-to-end transaction management for composite e-services. Furthermore, a model for monitoring and controlling e-services is proposed in Ref. [16]. Their investigations do not provide recommendation facilities to help the design of composite e-services. Additionally, existing recommender systems, e.g., Refs. [3,12,15] focus on recommending top N relevant items (documents or products) regarding a given item. However, this work advises the flow schema of a set of given items (e-services).

This work proposes a novel data mining approach to recommend the means of composing customer-selected e-services. To do so, the metadata of e-services can be described by adding attributes of e-services to Universal Description, Discovery, and Integration (UDDI) standard [20]. Thus, customer can search desired e-services with semantic conditions. Moreover, standard Unified Modeling Language (UML) [21] activity diagram and ECA rules [11] are adopted to model composite e-services. Furthermore, the data mining approach is used to discover the frequent predicate sets of e-services and the frequent ordering (flow) sets among the instances of composite e-services. Based on the mining result, the proposed platform can recommend the top N composite e-services for customers to support their decisions.

The remainder of this work is organized as follows. Section 2 illustrates the related work. Section 3 presents the novel functions of the proposed e-service platform. The e-service metadata is illustrated in Section 4. Section 5 models and defines composite e-services from the workflow viewpoint. Section 6 presents the mining and recommendation of composite e-services. Conclusions and future work are finally discussed in Section 7.

2. Related work

2.1. E-service and e-services platform

Hewlett-Packard (HP) investigates several technical details of e-services and also defines e-services as follows: an e-service is any asset that you make available via the Internet to drive new revenue streams or create new efficiencies [9]. Web service is another term that is used to define the services provided via the Internet. Web services and e-services are similar, but Web services place an emphasis on Web technologies. E-speak is an open software platform designed specifically for the development, deployment, and intelligent interaction of e-services. The platform consists of the e-speak Service Frame Specification, e-speak Service Engine, and the e-speak Trading Communities Edition [9]. WebSphere [10] is a Web service platform designed by IBM to implement and deploy Web services. WebSphere is capable of hosting Web services based on standards, such as Simple Object Access Protocol (SOAP) [18], Web Services Description Language (WSDL) [24], and Universal Description, Discovery and Integration (UDDI) [20].

2.2. XML and WEB services standards

Extensible Markup Language (XML) [27] has been widely used in electronic commerce, as it uses a flexible, open, and standard-based format to provide interoperability of data exchange over the Internet. The impact and implementation of XML on business-to-business commerce has been discussed in Ref. [29]. SOAP [18], WSDL [24], and UDDI [20] have been accepted as the de facto standards for Web services. UDDI provides directory services for registering and searching e-services. WSDL is an XML-based language used to describe the usage (behavior) of e-services. SOAP is an XML-based protocol for exchanging request/response messages between e-services providers and customers. The roles of these three standards are indicated in Fig. 1. Detailed introductions of these standards can be found in Refs. [8,19].

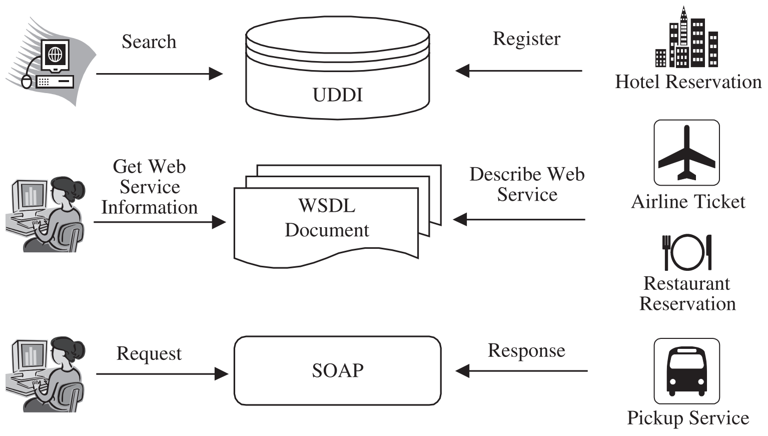


Fig. 1. UDDI, WSDL, and SOAP for Web services.

2.3. Composite e-services

Composite e-services are composed by several e-services, and, intuitively, it can be seen as a workflow. A workflow is an automated business process that manages the sequence of work activities and the use of appropriate human or IT resources associated with the various activity steps. Casati and Shan [6,7] proposed a dynamic model for composing e-services. Piccinelli et al. [14] indicated that the semantics and characteristics of an e-service are described with an ontology-based approach and are encoded into XML documents. Piccinelli et al. also present a DySCo infrastructure to model an e-service environment.

Some vendor protocols for constructing composite e-services are available, such as IBM's Web Service Flow Language (WSFL) [25] and Micorsoft's XLANG [26]. Recently, IBM, Microsoft, and BEA Systems merged their flow languages into "Business Process Execution Language for Web Services (BPEL4WS)" protocol [5]. There is also a similar protocol: Web Service Choreography Interface (WSCI) [23], co-developed by BEA Systems, Intalio, SAP, and Sun Microsystems.

2.4. Process mining

Workflow design is complex and time-consuming. Therefore, process mining is widely used in discovering the actual process models. Process mining approaches [1,22] mainly use workflow logs to discover the workflow process as it is actually being executed. Weijters and Van der Aalst [22] proposed a process mining approach and distinguished three mining steps. The first step is the construction of a dependency/frequency table (D/F-table) that can acquire the dependencies between the activities. Next, a D/F-graph can be drawn from the D/F-table. Finally, the workflow model is discovered after determining AND-splits and XOR-splits. Agrawal et al. [1] proposed algorithms to discover workflow models. They also consider transitive reduction and other noises into the process mining algorithms.

Unlike most researches in process mining [1,22], the flow schemas of composite e-services are known. The flow schema is defined in Section 6. In this research, we use association rules to find frequent item sets from instance execution logs. We use the support value to give each flow schema a score and to find the top N flows for recommendations to customers.

3. System overview

Generally, an e-service platform provides various mediating facilities for e-service providers and customers. For e-services providers, an e-service platform is capable of registering e-services descriptions, advertising e-services in directories, monitoring e-services, and managing e-services. For customers, an e-services platform is capable of searching proper e-services and accessing e-services. Moreover, UDDI is employed to provide directory services for registering and searching e-services. WSDL is used to describe the usage of e-services, while SOAP is employed to exchange request/response messages between e-service providers and customers. This work proposes three enhancements to conventional e-service platforms, including design of e-service metadata, modeling of composite e-services, and recommendations of predicates of each e-service and orderings between e-services and composite e-services.

First, we design an e-service metadata for semantic search. Without e-service metadata, customers cannot conduct semantic searches such as discovering a hotel reservation e-service within NT $5000. The designed e-service metadata is presented in Section 4. We extend UDDI businessService type with e-service metadata. The metadata allows customers to discover desired e-services based on e-service attributes.

Second, we support e-service composition for delivering value-added e-service. Composite e-services, composed by several e-services, can be viewed as a workflow which prescribes the ordering of e-services. We use UML activity diagrams and ECA rules to describe the flow schema of composite e-services.

Finally, we propose a data mining approach for recommending composite e-services. When an e-service has been registered and advertised by a provider, the customers can query the e-service platform to discover interested e-services. Through mining instance execution logs and referring to flow schemas, the extended e-service platform can recommend the ways of composing the discovered e-services. Provided recommendations include the ordering between e-services, the selection conditions of e-services, and the top *N* matching composite e-service definitions.

The architecture view of the proposed e-service platform is shown in Fig. 2, which indicates the components, outbound standards, and users of the platform. Novel features are described below. Besides conventional search, the search tools adopt metadata of e-services to provide semantic search. Recommender is responsible for recommending the top *N* composite e-services according to the mining results provided by data mining tools. E-Service composition is handled by composite e-service definition tool and engine; the former defines and stores composite e-service definitions in design-time, while the latter manages composite e-service instances (that is, an

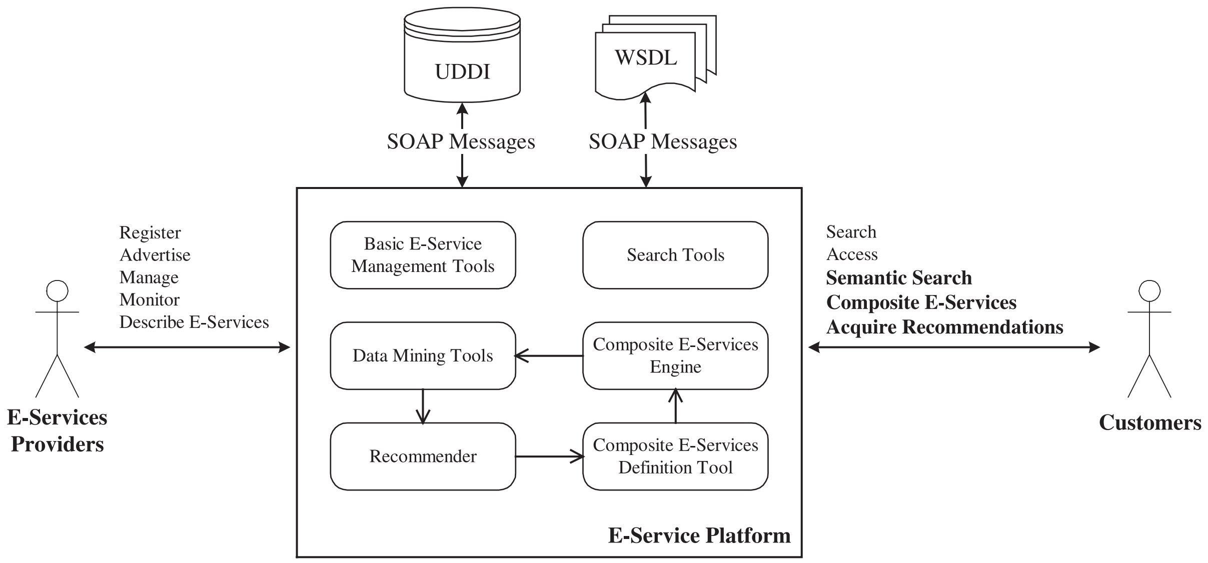


Fig. 2. Architecture of proposed e-service platform.

execution of a composite e-service definition) in runtime.

4. E-service metadata

4.1. Metadata design

E-service metadata refers to a detailed description about e-services and providers; it is used to advertise or discover e-services in registries. The metadata consists of two types of metadata: business and service level metadata. Business level metadata refers to the description of e-service providers, while service level metadata describes the detailed characteristics of e-services. E-services metadata benefits both e-service providers and customers. Namely, providers can advertise their e-service more precisely, while customers can discover desired e-services and providers more efficiently. To facilitate the discovery and advertisement of e-services and providers, the enhancement of business and service level metadata are proposed as follows.

4.1.1. Business level metadata

Business level metadata describes the basic information about e-service providers, such as name, phone number, address, and e-mail. UDDI is a repository, which contains descriptions of e-services and their providers. The businessEntity [20], one type of UDDI, contains all known information about a business or entity that publishes descriptive information about the entity as well as the services that it offers. Therefore, the data structure of businessEntity can be used to describe the business level metadata.

Fig. 3 shows an example that uses UDDI BusinessEntity as business level metadata for describing an e-service provider. It shows the discoveryURL, name, description, and contact information of "Paradise Company".

4.1.2. Service level metadata

The service level metadata contains the characteristics that can describe the detailed information about the e-services. Through the service level metadata, customers can identify the content and fulfillment conditions of available e-services. The businessService structure of UDDI represents a logical service classification [20], and only states the name, description, and binding information of an e-service. However, the information of businessService in UDDI can be extended to provide semantic search. We add extended e-service metadata as listed below in the businessService type of UDDI to keep compatible with existing standards. Fig. 4 shows a fragment of businessService type that contains the service level metadata.

ServiceCategory: ServiceCategory refers to high-level metadata. E-services that provide the same services will be grouped in a ServiceCategory. For instance, "hotel reservation" is one kind of ServiceCategory.

```
<BusinessEntity businessKey="…" >
    <discoveryURLs >
        <discovery URL >http://www.paradise.com.tw</discovery URL>
    </discoveryURLs>
    <name>Paradise</name>
    <description>………</description>
    <contacts useType="…">
          <personName>Joe Wang</personName>
          <phone>02-89765432</phone>
          <email> paradise@paradise.com</email>
         <address> No. 131, Sec. 3, Nanjing E. Rd., Sungshan Chiu, Taipei,
                   Taiwan, R.O.C. </address>
    </contacts>
    <businessServices> Hotel_Reservation </businessServices>
    <identifierBag>……</identifierBag>
    <categoryBag>……</categoryBag>
</BusinessEntity>
```

Fig. 3. Business level metadata (using BusinessEntity structure).

```
<businessService serviceKey="…" >
    <name>Hotel_Reservation</name>
    <description>
      <extend xmlns:serviceExt="…">
        <serviceExt:ServiceCategory> Hotel_Reservation
        </serviceExt:ServiceCategory>
        <serviceExt:LocatedNear> Downtown </serviceExt:LocatedNear>
        <serviceExt:ServiceLocation> Taiwan Taipei
        </serviceExt:ServiceLocation>
        <serviceExt:Facility> swimming pool </serviceExt:Facility>
        <serviceExt:QualityRating> five stars </serviceExt:QualityRating>
        <serviceExt:Guarantees> breakfast </serviceExt:Guarantees>
```

Fig. 4. Embed the service-level metadata in UDDI businessService.

ServiceLocation: ServiceLocation refers to the place of the service. A person who wants to find a hotel in America cannot reserve a hotel in France.

LocatedNear: LocatedNear refers to the geographic area of the service. The attribute can be "downtown", "suburb", and "airport".

Facility: Facility refers to special offers by the service. For example, a customer may want to find a hotel with a swimming pool.

QualityRating: QualityRating refers to an expandable list of rating properties that may accompany a service. For example, quality rating in hotel reservation may be presented by five stars.

Guarantees: Guarantees refer to promises of services. A hotel reservation may guarantee that breakfast is included in the service. A delivery service may guarantee delivery in 24 h.

TotalCapacity: TotalCapacity refers to the total numbers the service can provide. A person who

(a) find_service request in UDDI

```
<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV=http://schemas.xmlsoap.org/soap/envelope/
                   xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
                   xmlns:xsd="http://www.w3.org/1999/XMLSchema">
<SOAP-ENV:Body>
     <find_service generic="1.0" xmlns="urn:uddi-org:api">
         <name>Hotel_Reservation</name>
     </find_service>
 </SOAP-ENV:Body>
 </SOAP-ENV:Envelope>
```

(b) find_service response in UDDI

```
<serviceList generic="2.0" operator="…" …>
      <serviceInfos>
            <serviceInfo serviceKey="3D45…" businessKey="2E4C…">
                 <name>Hotel_Reservation</name>
            </serviceInfo>
      </serviceInfos>
</serviceList>
```

(c) semantic search in XQuery

```
<result>
   for $es in EZTravel-UDDI//businessService
   where $es//serviceExt:ServiceCategory = "Hotel_Reservation" and
      $es//serviceExt:LocatedNear = "Downtown" and
      $es//serviceExt:Facility = "Swimming Pool" and
      $es//serviceExt:QualityRating = "5 stars"
   return $es
</result>
```

Fig. 5. Service discovery.

wants to reserve 100 hotel rooms will not reserve a small hotel with only 50 rooms.

OfferCapacity: OfferCapacity refers to the last numbers that the service can provide. The value of OfferCapacity dynamically changes according to the capacity that can be offered.

Price: Price refers to the cost per service.

4.2. Using metadata for semantic search

UDDI provides find_business and find_service APIs to search for businesses and services, respectively [20]. The extended metadata, added in businessService, can be searched by XQuery [28] statements. The UDDI find_business API requests and returns a businessList message that matches the specific conditions. For example, Fig. 5a and b show the UDDI find_service API requests and responses with specific services. The UDDI find_service requests an e-service with the name "Hotel_Reservation". The UDDI find_service responds with e-service information whose name is "Hotel_Reservation".

With the support of e-service metadata, customers can conduct semantic search according to e-service content. Taking "hotel reservation", for example, the predicates can be {LocatedNear = Downtown, Facility = Swimming Pool, QualityRating = 5 stars} or {LocatedNear = Suburb, QualityRating = 3 stars}. Predicates are expressed in XQuery. We can make a query to select proper e-services whose metadata information matches the predicates. Fig. 5c is an example of semantic search, and the query responses

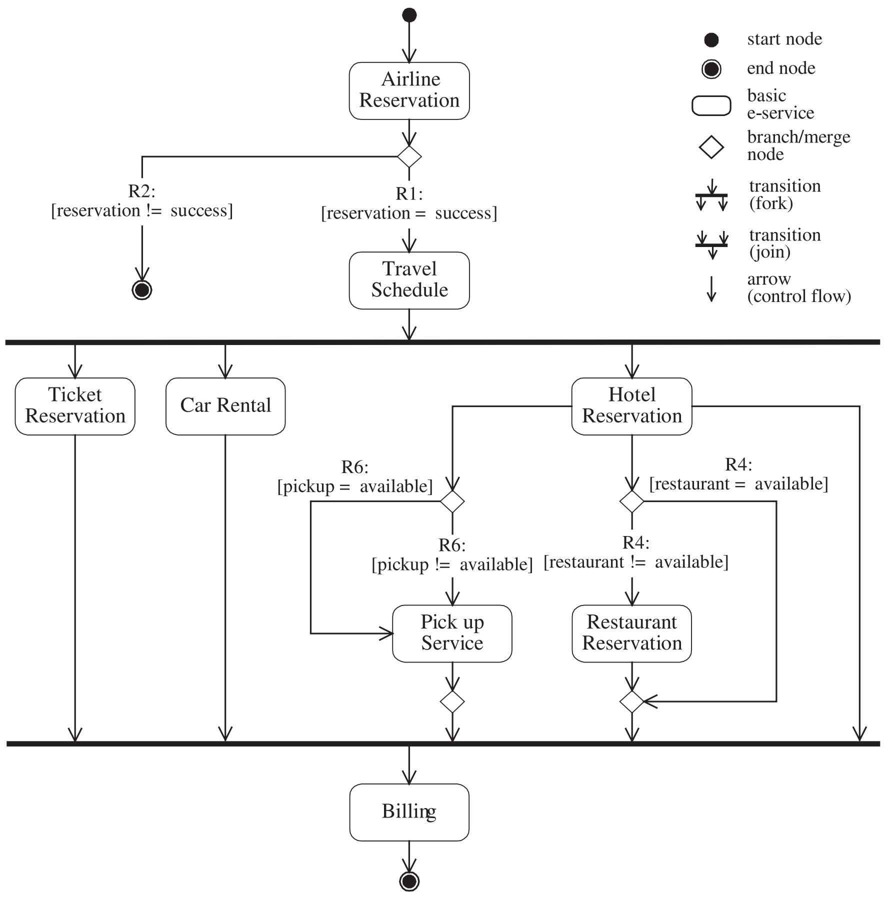


Fig. 6. Composite e-service example.

Table 1
Syntax of routing rules and e-Service selection rules

(a) Routing rule (**R**)	(b) E-service selection rule (**SR**)
E: Events	E: Events
C: Conditions	C: Conditions
A: Actions	A: Actions
Define Rule R	Define Rule SR
On ⟨**Events**⟩ Do	On ⟨**Events**⟩ Do
If ⟨**Conditions**⟩ is True Then	If ⟨**Conditions**⟩ is True Then
Execute ⟨**Actions**⟩	Execute ⟨**Actions**⟩
Activated for **preceding Basic E-Service**	Activated for **Notify E-Service Provider**

with five-star hotels that are located downtown and have swimming pools.

5. Composite e-services

A composite e-service, composed by several e-services, is similar to a workflow. A workflow specifies the ordering of tasks and is generally represented as a directed graph. Similarly, a composite e-service coordinates the enactment sequence of member e-services. We apply UML activity diagram to describe the flow schema of a composite e-service. Additionally, Event/Condition/Action (ECA) rule provides a flexible event-driven manner to trigger the enactment of activities and select activity perforptmers during workflow run-time. Thus, we use ECA rules to control the e-service enactment and select e-service providers.

5.1. Flow schema of composite e-services

The flow schema of a composite e-service should be defined to describe the enactment sequence of its basic e-services. Fig. 6 shows the flow schema of a composite travel e-service as represented by an activity diagram.

As representing an activity in a workflow, an activity state in an activity diagram denotes a basic e-service in the flow schema of a composite e-service. Basic e-services are the e-services provided on the Internet to access. Different e-service providers can provide the same basic e-service. Additionally, a start node represents the beginning of this flow, while an end node represents the completion of this flow. A branch node is a set of transitions leaving a single state such that exactly one guard condition on one of the transitions must always be satisfied. Arrows represent the flow dependencies and ordering between e-services. A composite e-services instance is an enactment of a flow schema of a composite e-service. A composite e-service can be instantiated several times.

5.2. ECA rules

Event/Condition/Action (ECA) rules have been widely used in workflow management systems as an activity scheduler [11]. This work uses ECA rules to control the routing of basic activities and the selection of e-service providers.

5.2.1. Using ECA rules for routing rules

When the input arrow is fired, a fork node fires all the output arrows in parallel, while a branch node fires the output arrows that satisfy the routing conditions. ECA rules can be used to describe the routing decisions. The syntax of routing rules is illustrated in Table 1a. In composite e-services, the completion of preceding e-services is regarded as an event of routing rules. For example, Table 2 shows two routing rules, R1 and R2, derived from Fig. 6. The routing rule R1 is fired after the completion of Airline_Reservation and the reservation is successful. R1 then notifies a travel schedule provider to prepare Travel_Schedule e-service. Contrarily, R2 is fired if the reservation is failed. Consequently, the composite e-service is ended.

Table 2
Routing rules of composite travel e-services

R1	R2
Define Rule R1	Define Rule R2
On **Airline Reservation completed** Do	On **Airline Reservation completed** Do
If **reservation** = **success** is True Then	If **reservation !=** **success** is True Then
Execute**Travel_Schedule_Service; Notify_Travel_Schedule_Provider**	Execute **End**
Activated for Airline Reservation	Activated for Airline Reservation
Reservation = **success** depends on the AirlineReservation: OfferCapacity; if the OfferCapacity is enough then **Reservation** = **success**	**Reservation !** = **success** depends on the AirlineReservation: OfferCapacity; if the OfferCapacity is not enough then **Reservation !** = **success**

Table 3
E-service selection rules

SR1: Airline Reservation Selection	**SR2: Travel Schedule Selection**
Define Rule SR1	Define Rule SR2
On **Pre Notify_Airline_Reservation_Provider** Do	On **Pre Notify_Travel_Schedule_Provider** Do
If **CustomerSelection** is True Then	If **SystemSelection** is True Then
Execute **XQuery for Airline Reserv. Provi. Selection;**	Execute **XQuery for Travel Schedule Provi. Selection;**
Add: Qualified_Providers; CustomerSelection	**Add: Qualified_Providers; SystemSelection**
Activated for **Notify_Airline_Reservation_Provider**	Activated for **Notify_Travel_Schedule_Provider**
XQuery for Airline Reserv. providers selection:	**XQuery for Travel Schedule providers selection:**
for $esp in EZTravel – UDDI//businessEntity	For $esp in EZTravel-UDDI//businessEntity
where $es//serviceExt:ServiceCategory	where $esp//serviceExt:ServiceCategory
= “Airline_Reservation” and	= “Travel_Schedule” and
$esp//serviceExt:Facility = “Personal Monitor” and	$esp//serviceExt:Facility = “Personal Monitor” and
$esp//serviceExt:Price = 500	$esp//serviceExt:Price = 50
return $esp	return $esp
Selection Policy: CustomerSelection	Selection Policy: SystemSelection

5.2.2. Using ECA rules for e-service selection rules

We can also use ECA rules as e-service selection rules. Each e-service associates with an e-service selection rule that determines an e-service provider. Table 1b is the syntax of e-service selection rules, and Table 3 shows the e-service selection rules of basic e-service in Fig. 6. Take SR2 for example, before notifying a travel schedule provider, SR2 executes an XQuery to identify the Qualified_Providers, that is, the providers who fit the query predicates.

An e-service selection rule executes an XQuery to discover several e-service providers who satisfy the predicates. The service selection rule further needs a selection policy that is capable of selecting one from the qualified e-service providers during run-time. Two kinds of selection policies are defined: CustomerSelection and SystemSelection. CustomerSelection policy requests customers to determine the e-service provider, while SystemSelection policy randomly selects the e-service provider by the system. For example, if the Qualified_Providers are A, B, C in SR1, a customer can select A to provide Airline Reservation. If the Qualified_Providers are D, E, F in SR2, the system can randomly select one of them.

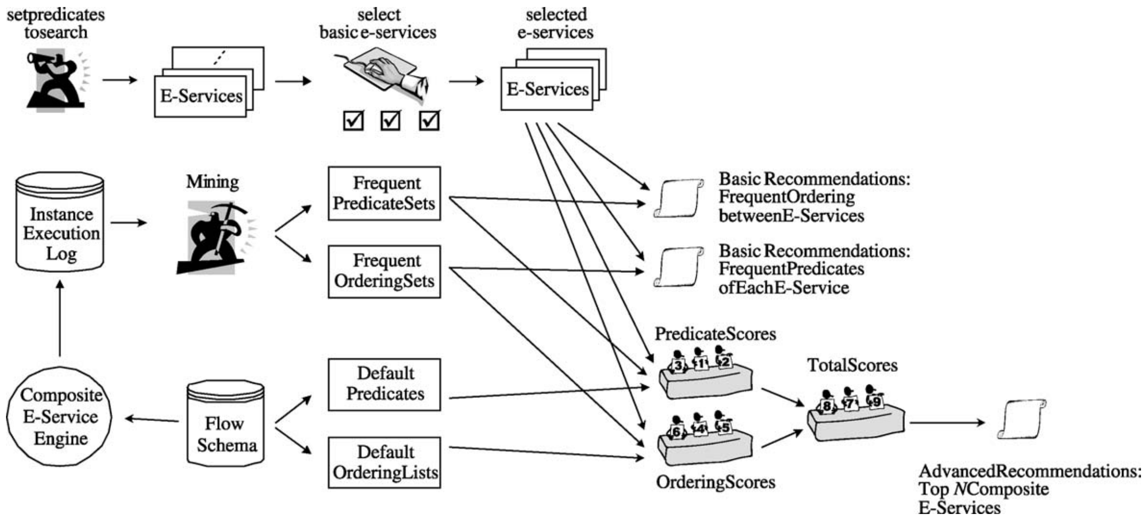


Fig. 7. Recommendations for customer-selected e-services.

6. Recommend composite e-services

The customer must specify the ordering and predicates of selected e-services to define a composite e-service. This work proposes a novel mining approach to simplify the composition step. The extended e-service platform can recommend the predicates of each basic e-service, the ordering between e-services, and the top *N* matching composite e-services that contain the desired e-services. Fig. 7 shows the proposed recommendation process for selected e-services. The Flow Schema Database stores the existing composite e-service definitions. The Instance Execution Log Database records previous executions of flow schema definitions. A flow schema may be instanced several times. Each instance can be stored as a log.

This work uses a data mining approach to acquire frequent predicate sets of each basic e-service from the Instance Execution Log Database. The frequent attributes of basic e-services are termed as frequent predicate sets in this work. This work also acquires frequent ordering sets from the Instance Execution Log Database. The frequent orderings between e-services are termed as frequent ordering sets. To provide advanced recommendations of complete composite e-services, this work uses a scoring approach to recommend the top *N* composite e-services for customers. The extended e-services platform can compute a frequent ordering score, according to the selected e-services, frequent ordering sets, and flow schema of composite e-services in the Flow Schema Database. Similarly, the system can compute a frequent predicate score, according to the selected e-services, frequent predicate sets, and default predicates of each e-service in the Flow Schema Database. A total score can be derived by summing up the frequent predicate score and the frequent ordering score. Finally, the platform recommends the top *N* composite e-services based on the ranking of total scores.

6.1. Mining instances of composite e-services

Unlike conventional process mining investigations [1,22], which focus on discovering previously unknown process models, the flow schema definitions of the instances are given conditions in this work. We use Apriori algorithm [2] to find frequent predicate sets and frequent ordering sets from instance execution logs. We also use the support value of frequent predicate and ordering sets to rate each composite e-service for recommending the top *N* composite e-services.

6.1.1. Mining frequent predicate sets

Assume that a composite travel e-service contains four basic e-services: airline ticket reservation, hotel reservation, travel schedule, and restaurant reservation. The system can recommend the frequent predicate sets of each basic e-service. The following illustrates the mining of frequent predicate sets of hotel reservation. The frequent predicates of other e-services are derived similarly.

Table 4 lists the predicates of the hotel reservation e-service. CID C001,. . ., and C024 are composite e-services instances that include the hotel reservation e-service. CID is the identifier of composite e-service instances in the instance execution logs. According to

Table 4
Instance execution logs (including hotel reservation)

CID	List of Predicates (Hotel Reservation)
C001	LocatedNear = Downtown, Facility = Swimming Pool, QualityRating = 5 stars
C002	LocatedNear = Downtown, Facility = Swimming Pool, QualityRating = 5 stars
C003	LocatedNear = Suburb, QualityRating = 3 stars
C004	LocatedNear = Downtown, Facility = Swimming Pool, QualityRating = 5 stars
C009	LocatedNear = Downtown, QualityRating = 3 stars
C010	LocatedNear = Suburb, Facility = Swimming Pool, QualityRating = 5 stars
C012	LocatedNear = Suburb, QualityRating = 3 stars
C015	LocatedNear = Suburb, QualityRating = 4 stars
C016	LocatedNear = Downtown, QualityRating = 4 stars
C017	LocatedNear = Suburb, Facility = Swimming Pool, QualityRating = 4 stars
C019	LocatedNear = Downtown, QualityRating = 3 stars
C024	LocatedNear = Suburb, Facility = Swimming Pool, QualityRating = 5 stars

Table 5
Frequent predicate sets of hotel reservation (min_sup = 25%)

Frequent Predicate Sets	sup
LocatedNear = Downtown, Facility = Swimming Pool, QualityRating = 5 stars	25%
Facility = Swimming Pool, QualityRating = 5 stars	42%
LocatedNear = Downtown, Facility = Swimming Pool	25%
LocatedNear = Downtown, QualityRating = 5 stars	25%
LocatedNear = Suburb, Facility = Swimming Pool	25%
LocatedNear = Downtown	50%
LocatedNear = Suburb	50%
Facility = Swimming Pool	50%
QualityRating = 5 stars	42%
QualityRating = 3 stars	33%
QualityRating = 4 stars	25%

these log data, we can use the Apriori algorithm [2] to discover the frequent predicate sets. *Support* of a predicate set PS, denoted by sup(PS), is the ratio of those instances that contain all predicates in PS. Suppose that the required minimum support value is 25%, predicate sets with support values greater than minimum support value are called *frequent* predicate sets. Table 5 shows the frequent predicate sets and their corresponding support values.

6.1.2. Mining frequent ordering sets

The following illustrates the derivation of a frequent ordering set. For brevity, we omit the start/end node and use a capitalized letter to represent a basic e-service in a composite e-service graph, as shown in Fig. 8. Notably, the *ordering* between e-services A and C, denoted by ⟨A, C⟩, means that A precedes C in the composite e-services. Obviously, if there exists a path from A to C, then A precedes C. Therefore, we can derive the set of ordering list: {⟨A, B⟩ ⟨A, C⟩ ⟨A, D⟩ ⟨A, E⟩ ⟨B, D⟩ ⟨B, E⟩ ⟨C, D⟩ ⟨C, E⟩ ⟨D, E⟩} from the composite e-service described in Fig. 8.

The ordering lists can be acquired according to the instance execution log and flow schema definitions, as partially shown in Table 6. CSID are the identifiers of flow schema definitions and CID are the identifiers of instances of flow schemas. Considering C001 whose instance execution log of C001 is ACD, we can get the set of ordering list, {⟨A, C⟩ ⟨A, D⟩ ⟨C, D⟩}. For another example, consider C004 whose instance execution log is BDCE. We may think that the ordering between D and C is ⟨D, C⟩. However, by referring to the definition of flow schema CS04, D and C are connected by a Fork node. That is, D and C are executed in parallel. Therefore, CS04 does not prescribe the ordering between D and C.

Every element ⟨X, Y⟩ in the ordering list is a candidate item for deriving frequent ordering sets. The support of an ordering set, *OS*, is the ratio of instances that contain all orderings in OS. Ordering sets with support values greater than the required minimum support values are called *frequent* ordering sets. We also use the Apriori Algorithm [2] to generate the frequent ordering sets that satisfy the required minimum support value. Suppose that the required minimum support value is 25%. Table 7 lists the frequent ordering sets and their corresponding support values.

6.2. Basic recommendations

A scenario is presented to illustrate the basic recommendations. Assume a customer wants to define a new composite travel e-service. He/she may select several travel e-services from the travel services pool. For example, the customer selects hotel reservation, airline reservation, travel schedule, and restaurant reservation. The extended e-services

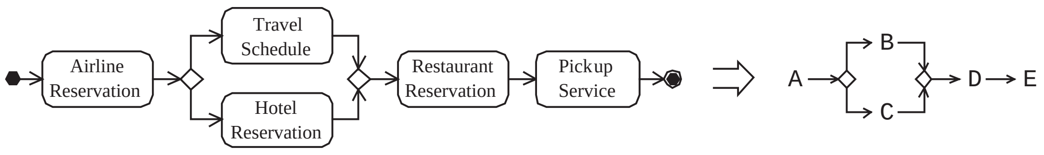
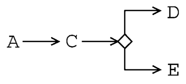
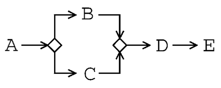
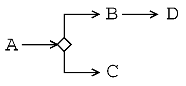
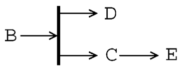


Fig. 8. Transfer composite e-services to a symbol diagram.

Table 6
Ordering list of composite e-services log

CSID	Flow Schema Definition	CID	Instance Execution Log	Ordering List
CS01		C001	ACD	⟨A,C⟩ ⟨A,D⟩ ⟨C,D⟩
		C003	ACE	⟨A,C⟩ ⟨C,E⟩ ⟨A,E⟩
CS02		C006	ABDE	⟨A,B⟩ ⟨A,D⟩ ⟨A,E⟩ ⟨B,D⟩ ⟨B,E⟩ ⟨D,E⟩
		C010	ACDE	⟨A,C⟩ ⟨A,D⟩ ⟨A,E⟩ ⟨C,D⟩ ⟨C,E⟩ ⟨D,E⟩
CS03		C007	ABD	⟨A,B⟩ ⟨A,D⟩ ⟨B,D⟩
CS04		C004	BDCE	⟨B,D⟩ ⟨B,C⟩ ⟨B,E⟩ ⟨C,E⟩
CS05	A→B→C→D	C002	ABCD	⟨A,B⟩ ⟨A,C⟩ ⟨A,D⟩ ⟨B,C⟩ ⟨B,D⟩ ⟨C,D⟩
			⋮	

platform should then recommend the predicates and orderings of these e-services.

According to the mining results in Section 6.1.1, the system acquires the frequent predicate sets of each basic e-service. The system can recommend these frequent predicate sets to the customers. For example, the system recommends a five-star hotel that is located near downtown with a swimming pool. According to the mining results in Section 6.1.2, the system acquires the frequent ordering sets between e-services. For example, according to Table 7, the ordering of airline reservation is greater than hotel reservation, travel schedule, and restaurant reservation. Moreover, the ordering of hotel reservation is greater than restaurant reservation. These frequent orderings are recommended to customers.

Table 7
Frequent ordering sets

Frequent ordering sets	sup
⟨A,B⟩ ⟨B,D⟩ ⟨A,D⟩	37%
⟨A,B⟩ ⟨A,D⟩	43%
⟨A,B⟩ ⟨B,D⟩	37%
⟨A,D⟩ ⟨B,D⟩	37%
⟨A,C⟩ ⟨A,D⟩	26%
⟨A,D⟩	60%
⟨A,B⟩	47%
⟨A,C⟩	40%
⟨B,D⟩	37%

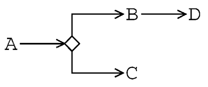
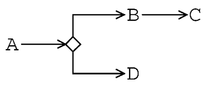
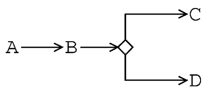
6.3. Advanced recommendations

Basic recommendations cannot suggest a complete flow of composite e-services. A customer needs advanced recommendations about complete composite e-services. This work uses a scoring approach on advanced recommendations. Two scores, a predicate score and an ordering score, are computed and mentioned in the following sections. The Flow Schema Database records the existing composite e-service definitions. We first extract the composite e-services that include the selected e-services. In this example, we extract the composite e-services that include e-services A, B, C, and D. Section 6.3.1 illustrates the computation of a predicate score for each extracted composite e-service. Section 6.3.2 illustrates the computation of an ordering score for each extracted composite e-service. The total score of a composite e-service is the summation of its predicate and ordering scores. Finally, the top *N* composite e-services are recommended to customers according to the total scores.

6.3.1. Predicate score

The system gives each composite e-service a predicate score. In this example, we compute the predicate score of each flow schema of composite e-service that includes A, B, C, and D. To consider the constraints of customers, the system may not only select the flow schemas which include A, B,

Table 8
Predicate, ordering, and total scores

CSID	Flow Schema Definition	A	B	C	D	Predicate Score	⟨A, B⟩	⟨A, C⟩	⟨A, D⟩	⟨B, D⟩	Ordering Score	Total Score	Rank
CS03		50	142	30	25	247	47	40	60	37	184	431	2
CS05	$A \rightarrow B \rightarrow C \rightarrow D$	60	100	50	50	260	47	40	60	37	184	444	1
CS10		60	92	50	25	227	47	40	60	×	147	374	5
CS25		40	100	50	30	220	47	40	60	37	184	404	3
CS30	$B \rightarrow A \rightarrow C \rightarrow D$	25	142	50	25	242	×	40	60	37	137	379	4
						⋮							

C, and D, but also select the ones whose predicates meet the customer's constraints.

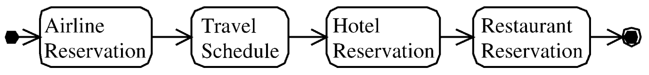
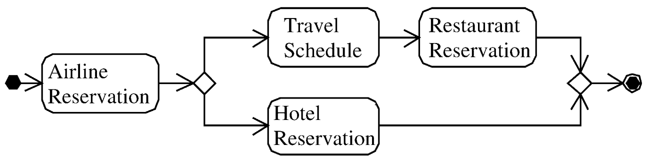
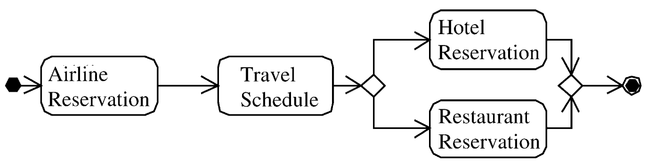
The predicate score of a composite e-service is the summation of the predicate scores of its basic e-services, as the following formula.

$$\text{Predicate Score of CS} = \sum_{e \in \mathrm{CS}} \sum_{p \in e.\text{predicates}} \sup(p)$$

where e denotes a basic e-service in a composite e-service CS; p represents a predicate in e. Currently, the predicate scores only consider the support values of predicates of e-services. Future work will consider the customer constraints.

Each e-service has default predicates in a flow schema definition. A default predicate can be a fixed value or a variable that is dynamically selected by a customer during run-time. Notably, for the variable predicates such as LocatedNear=%locate% and QualityRating=%rating%, sup(LocatedNear=%locate%) returns the maximum support between LocatedNear = Downtown and LocatedNear = Suburb, and sup(QualityRating=%rating%) returns the maximum

Table 9
Advanced recommendations

Recommended Composite E-Services	Rank
	1
	2
	3

support among QualityRating = 3 stars, QualityRating = 4 stars, and QualityRating = 5 stars. Table 8 shows the CSID, flow schema definition, and the predicate score of composite e-services.

6.3.2. Ordering score

The system also computes the ordering score of composite e-services that include A, B, C, and D. The Ordering score can be derived by the following formula.

$$\text{Ordering Score of CS} = \sum_{\langle x,y\rangle \in \text{CS.ordering}} \text{sup}(\langle x,y\rangle)$$

where $\langle x, y\rangle \in$CS.ordering means that the ordering $\langle x, y\rangle$ holds in the flow schema of the composite e-service CS. For example, in CS03, the supports of $\langle$A, B$\rangle$, $\langle$A, C$\rangle$, $\langle$A, D$\rangle$, and $\langle$B, D$\rangle$ are 47, 40, 60, and 37, respectively. Therefore, the ordering score of CS03 = sup($\langle$A, B$\rangle$) + sup($\langle$A, C$\rangle$) + sup($\langle$A, D$\rangle$) + sup($\langle$B, D$\rangle$) = 47 + 40 + 60 + 37 = 184. Ordering scores of CS05, CS10, CS25, and CS30 are derived in the same way. Table 8 shows the ordering scores of candidate composite e-services.

6.3.3. Total score

The total scores are derived by summing up the predicate scores and ordering scores, as shown in Table 8, i.e., Total Score = Predicate Score + Ordering Score. The system ranks extracted composite e-service based on the total score. Notably, a weighted score can also be computed by multiplying the scores with corresponding weights. That is, Total Score=($w_p \times$ Pre-Predicate Score)+($w_o \times$ Ordering Score), where w_p and w_o are the weights of predicate and ordering score, respectively.

6.3.4. Recommendations

Finally, the system recommends the top N flows of composite e-services to the customers. The top three recommended composite e-services are shown in Table 9, after converting to original representations.

7. Conclusions and future work

This work proposes novel enhancements in e-service discovery and composition. First, this work designs metadata of e-services that describes the characteristics of e-services. Customers can discover e-services through semantic predicates rather than content-independent queries, since providers can expose more meaningful description of e-services by using the proposed metadata.

Second, this work includes a workflow model to represent composite e-services closely like workflows. The model uses an activity diagram of UML to describe the flow schema of composite e-services, and provides an event-driven way to control the execution of basic e-services by using ECA rules. Moreover, this work analyzes instance execution logs by using association rules to discover frequent predicates of e-services and frequent orderings between e-services for basic recommendations. Furthermore, the mining results are used to score default flows of composite e-services. The top N composite e-services are discovered for advanced recommendations.

Further work will address two directions. This work uses a mining approach in recommending composite e-services. The mining of frequent ordering sets of e-services does not consider the conditions in routing rules. In the future, the conditions in routing rules should be considered to recommend composite e-services with routing rules. Besides, future work will construct user profiles (both customer and provider of e-services) to provide more personalized recommendations.

Acknowledgements

This research was supported by the National Science Council of the Republic of China under the grant NSC 91-2416-H-009-008.

References

[1] R. Agrawal, D. Gunopulos, F. Leymann, Mining process models from workflow logs, Proceedings of the 6th International Conference on Extending Database Technology (EDBT'98), Valencia, Spain, March 23–27, Springer-Verlag, Berlin, Germany, 1998, pp. 469–483.

[2] R. Agrawal, T. Imielinski, A.N. Swami, Mining association rules between sets of items in large databases, Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, DC, USA, May 26–28, ACM Press, New York, NY, USA, 1993, pp. 207–216.

[3] M. Balabanovic, Y. Shoham, FAB: content-based, collabora-

tive recommendation, Communications of the ACM 40 (3) (1997) 66–72.

[4] R. Balakrishnan, A service framework specification for dynamic e-services interaction, Proceedings of the 4th International Enterprise Distributed Objects Computing Conference (EDOC'00), Makuhari, Japan, September 25–28, IEEE Computer Society Press, Los Alamitos, CA, USA, 2000, pp. 28–37.

[5] BPEL4WS, Business Process Execution Language for Web Services, http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/.

[6] F. Casati, M.-C. Shan, Definition, execution, analysis, and optimization of composite e-services, IEEE Data Engineering Bulletin 24 (1) (2001) 29–34.

[7] F. Casati, M.-C. Shan, Dynamic and adaptive composition of e-services, Information Systems 26 (3) (2001) 143–163.

[8] F. Curbera, M. Duftler, R. Khalaf, W. Nagy, N. Mukhi, S. Weerawarana, Unraveling the Web services Web: an introduction to SOAP, WSDL, and UDDI, IEEE Internet Computing, (2002, Mar./Apr.) 86–93.

[9] Hewlett Packard (HP), E-Speak, http://www.e-speak.hp.com/.

[10] IBM, Websphere Application Server, http://www7b.boulder.ibm.com/wsdd/products/platformoverview.html.

[11] G. Kappel, P. Lang, S. Rausch-Schott, Workflow management based on objects, rules, and roles, IEEE Data Engineering Bulletin 18 (1) (1995) 11–18.

[12] J.A. Konstan, B.N. Miller, D. Maltz, J.L. Herlocker, L.R. Gordon, R. Riedl, GroupLens: applying collaborative filtering to usenet news, Communications of the ACM 40 (3) (1997) 77–87.

[13] Microsoft, Net, http://www.microsoft.com/net.

[14] G. Piccinelli, G.-D. Vitantonio, L. Mokrushin, Dynamic service aggregation in electronic marketplaces, Computer Networks 37 (2) (2001) 95–109.

[15] P. Resnick, H.R. Varian, Recommender systems, Communications of the ACM 40 (3) (1997) 56–58.

[16] A. Sahai, V. Machiraju, K. Wurster, Monitoring and controlling internet based e-services, Proceedings of the 2nd IEEE Workshop on Internet Applications (WIAPP'01), San Jose, CA, USA, July 23-24, IEEE Computer Society Press, Los Alamitos, CA, USA, 2001, pp. 41–48.

[17] A. Sahai, J. Ouyang, V. Machiraju, End-to-end transaction management for composite web based services, Proceedings of the 3rd International Workshop on Advanced Issues of E-Commerce and Web-Based Information Systems, San Jose, CA, USA, June 21–22, IEEE Computer Society Press, Los Alamitos, CA, USA, 2001, pp. 128–135.

[18] SOAP, Simple Object Access Protocol, http://www.w3.org/TR/soap.

[19] A. Tsalgatidou, T. Pilioura, An overview of standards and related technology in web services, Distributed and Parallel Databases 12 (2–3) (2002) 135–162.

[20] UDDI, Universal Description, Discovery and Integration, http://www.uddi.org/.

[21] UML, Unified Modeling Language, http://www.uml.org/.

[22] A.J.M.M. Weijters, W.M.P. Van der Aalst, Process mining: discovering workflow models from event-based data, Proceedings of the 13th Belgium-Netherlands Conference on Artificial Intelligence (BNAIC 2001), Amsterdam, Netherlands, October 25–26, University of Amsterdam, Amsterdam, The Netherlands, 2001, pp. 283–290.

[23] WSCI, Web Service Choreography Interface, http://wwws.sun.com/software/xml/developers/wsci/.

[24] WSDL, Web Services Description Language, http://www.w3.org/TR/wsdl.

[25] WSFL, Web Service Flow Language, http://www-3.ibm.com/software/solutions/webservices/pdf/WSFL.pdf.

[26] XLANG, Web Services for Business Process Design, http://www.gotdotnet.com/team/xml_wsspecs/xlang-c/.

[27] XML, Extensible Markup Language, World Wide Web Consortium (W3C) At URL: http://www.w3.org/XML.

[28] XQuery, XML Query, http://www.w3.org/XML/Query.

[29] D.C. Yen, S.-M. Huang, C.-Y. Ku, The impact and implementation of XML on business-to-business commerce, Computer Standards & Interfaces 24 (2002) 347–362.

Duen-Ren Liu received the BS and MS degrees in Computer Science and Information Engineering from the National Taiwan University, Taiwan, in 1985 and 1987, respectively, and the PhD degree in Computer Science from the University of Minnesota in 1995. He is currently an associate professor of the Institute of Information Management, National Chiao Tung University, Taiwan. His research interests include database systems, information systems, electronic commerce, workflow systems, and Internet applications. Dr. Liu is an associate member of the IEEE and a member of the ACM.

Minxin Shen is a PhD student at Institute of Information Management, National Chiao Tung University, Taiwan. He received his Bachelor's degree, with a double major in Business Administration and Computer Science, from Feng Chia University, Taiwan, in 1998. His current research interests include workflow management systems, computer supported cooperative work, web services, electronic commerce, and decision support technologies.

Chiu-Ting Liao received the BA degree in Information Management from the Fu Jen Catholic University, Taiwan, in 2000, and the MBA degree in Information Management from the National Chiao Tung University, Taiwan, in 2002. Currently, she is an engineer at the Information Technology Service division of BenQ. Her research interests include electronic commerce, information systems and workflow systems.