

# DYNAMIC AND PERSONALIZED WEB SERVICES COMPOSITION IN E-BUSINESS

Dongsong Zhang, Minder Chen, and Lina Zhou

**Building composite Web services can save significant time and cost for developing new applications and enhancing the interoperability and collaboration among E-business partners. This article proposes a framework for dynamic and personalized composition of Web services using an approach that integrates not only functional attributes of Web services, but also nonfunctional attributes such as service requirements, quality of service, and the preferences and constraints of individual service consumers. A prototype system based on the proposed framework and some implementation details are also presented.**

**S**ERVICE-ORIENTED COMPUTING (SOC) is a computing paradigm that uses services as fundamental elements of the development process of business applications. A growing trend in SOC is to build programming language- and platform-independent software components for Web applications. Today, companies are undertaking a shift toward tying their competitive advantages to a “dynamic ecosystem” of business Web services with their trading partners such as suppliers, distributors, agents, dealers, brokers, and retailers. Creating a collaborative network of trading partners to reduce costs, shorten business process cycles, and streamline workflows is critical to business competitiveness and growth. The emerging Web services technology provides a basis for the interaction and coordination of business processes distributed across different organizations. With Web services, new applications can be quickly developed by assembling existing individual applications. Some industries, such as insurance and travel, have already started to tap into the power of Web services, enabling them to integrate seamlessly with applications

of business partners and provide new services to their consumers. For example, Amazon.com has begun using XML and Web services to expose valuable company data to independent software developers. It is aimed to keep the barriers of using Web services as few as possible, while maximizing the amount of innovation.

One of the major challenges in Web services is the automated service composition in support of service-oriented processes. Current Web service techniques are primarily limited to services where each operation is independent. In many situations, however, a consumer’s request cannot be fulfilled by any existing single Web service. As a result, effective solutions for composing existing services into a composite Web service are being sought. The ability to plan Web service composition (WSC) and monitor execution of composite services automatically is an essential step toward the real usage of Web services (Pistore et al., 2004).

Another challenge in Web services is personalization. So far, most Web services are provided to different consumers in exactly the same way. For example, in the real estate industry, home

*DONGSONG ZHANG (zhangd@umbc.edu) and LINA ZHOU (zhoul@umbc.edu) are assistant professors in the Department of Information Systems at the University of Maryland, Baltimore.*

*MINDER CHEN (mchen@gmu.edu) is an associate professor in the MIS and Decision Science Department of the School of Management at George Mason University.*

***This article proposes a generic framework for dynamic composition and personalization of Web services.***

buyers use an online home listing service to look for houses on the market. Due to different needs and preferences, however, individual buyers' interests in houses vary in location, price range, size of the house, and other related factors. It is highly desirable for home listing service providers to present house listings that are automatically customized for individual buyers. Personalized Web services refer to those services that can be customized or tailored to the needs and preferences of individual service consumers. So far, little research has been done on the personalization of Web services.

To address the above problems, this article proposes a generic framework for dynamic composition and personalization of Web services and discusses various enabling techniques in detail. In particular, an innovative approach to Web service composition planning that integrates functional and nonfunctional attributes of Web services, as well as preferences of individual consumers, is presented. The remainder of the article is organized as follows. After introducing the concepts of Web services and service composition, we briefly describe dynamic and personalized WSC. Then we present the proposed framework, which incorporates a three-stage approach to the selection of component Web services. The framework is then illustrated with a prototype real estate business application.

## **WEB SERVICES AND SERVICE COMPOSITION**

### **Web Services**

Defined by the World Wide Web Consortium (W3C), a Web service is a software component identified by a URL, whose public interfaces and bindings are defined and described using XML (eXtensible Markup Language). Gartner (<http://www3.gartner.com>) defines a Web service as a custom end-to-end application that interoperates with other commercial and custom software through a family of XML interfaces to perform useful business functions. Web services are independent of any programming language, location, platform, and object model.

Web services provide a standard-based approach to implementing distributed software components by offering data and business logic services over standards that describe a service-oriented, component-based application architecture. The Web service standards stack includes Internet protocols (i.e., HTTP) for networking, XML for data format, SOAP (Simple Object Access Protocol) for messaging, WSDL

(Web Services Description Language) for service definition, and UDDI (Universal Description, Discovery, and Integration) for service registration. To be able to operate in an SOC environment, Web services must declaratively define their properties in a standard and machine understandable format. WSDL is an XML-based language. It specifies a Web service by defining messages that provide an abstract definition of the data being transmitted and operations that the service provides.

There are three roles in the Web service architecture: the service provider, consumer, and service registry. Interactions among these roles include service publishing, finding, binding, and invocation. A Web service provider publishes a Web service defined by WSDL in a service registry (e.g., UDDI). A service consumer can search and retrieve a service description and the access point to a WSDL file from the registry. The WSDL file is then used to generate a client-side Web service proxy that is used by the service consumer to invoke the Web service by sending a SOAP message. After the service is executed, the results will be returned to the service consumer as a SOAP message. SOAP enables communication between two applications that are developed in different programming languages and run on different platforms.

### **Web Service Composition**

Individual Web services cannot satisfy all the service requests. When that happens, it is desired to seek possibilities of combining existing services together to fulfill the request. A typical example of a service request is to place an order to an online retailer through a Web site. It can be fulfilled by a composite service that integrates several internal and external services such as credit checking service, inventory status-checking service (e.g., if the ordered product is out of stock, then another ordering service must be invoked from this retailer to potential suppliers to replenish its inventory), inventory update service, shipping service, etc.

WSC is to construct higher-level services based on multiple existing services to meet more sophisticated business requirements. Composing existing Web services into advanced, complex new services promotes rapid application development, service reuse, and cross-enterprise collaboration (Zhou and Zhang, 2004). WSC has the potential to reduce development time and effort for new applications. It provides a mechanism to support

***The ability to perform automated or semi-automated Web service composition (WSC) can revolutionize many application areas, including E-commerce applications and system integration.***

cross-enterprise and intra-enterprise application integration. Because it can be time- and cost-prohibitive to identify and compose services manually, automatic or semi-automatic composition of existing Web services to achieve new functionality has become the center of current attention. From a developer's perspective, service composition offers the possibility of interaction and reuse. From a service consumer's perspective, automated composition offers access to a wide variety of complex services with minimal manual intervention. The ability to perform automated or semi-automated WSC can revolutionize many application areas, including E-commerce applications and system integration.

A fundamental problem in WSC is to identify the most appropriate component services in order to generate an optimal composition plan. There are two generic approaches to automated Web service composition (Zhang, 2005). One is to define a composition at design time (called static service composition); the other is to plan service compositions and execute them dynamically at runtime (called dynamic service composition). There are several problems with the former approach. First, there may be a large number of services that provide the same business function. Second, new services may be created or existing services may be updated at any time. Third, the availability or response time of services may change over time. Fourth, there is dynamic variety among service offerings. Services may differ from each other on various factors ranging from the products they carry, prices, constraints, discounts, and after-sales support. Finally, service consumers may change their needs over time, which can affect the component services in a service plan. These factors should be taken into consideration while evaluating a potential service plan for a specific service request. Therefore, dynamic Web service composition at runtime is highly preferred (Rao et al., 2003). Such a plug-and-play service composition scheme significantly reduces the effort involved in the development and management of complex services, increases the quality of service composition, and is a critical step toward interactive, flexible, and collaborative E-businesses (Orriens et al., 2003). In SOA-based computing, the real power of Web services will be fully realized when diverse applications can be automatically integrated on-the-fly. However, due to the heterogeneity and dynamic nature of Web services, there are a lot of challenges and issues that need to be addressed to achieve this

ultimate goal. For example, the providers that provide the same services in a specific community should agree to follow certain requirements or standards for interface definition, so that a composite service can directly call any of those individual services in real-time, regardless of their actual implementations.

A number of standards have been developed to support WSC. They focus on representing Web service compositions where the process flow and bindings between services are known *a priori* (Rao et al., 2003). Web Services Flow Language (WSFL) is proposed to describe compositions of services in the form of a workflow. BPEL4WS (Business Process Execution Language for Web Services) is a standard for the specification and execution of service-oriented business processes. It provides programming-language-like constructs (e.g., while, pick, switch, and sequence) and graph-based links that represent ordering constraints on the constructs. BPEL4WS is designed with two functions in mind. Executable BPEL4WS programs allow the specification and execution of internal processes within an organization. Abstract BPEL4WS specifications are used to specify and publish invocations of and interactions with external Web services. The service platform functions like a workflow execution engine, performing and coordinating tasks of dispatching service calls, mediating and transforming intermediary data, and executing a service plan (Madhusudan and Uttamsingh, 2004). Strategies to cope with a variety of failures that may occur during the execution of a service plan are encoded in the service definition in advance, which cannot cover all possible scenarios in real-time service execution.

## **DYNAMIC AND PERSONALIZED WEB SERVICE COMPOSITION**

### **Dynamic Web Service Composition**

Although dynamic selection and composition of component services at runtime have been put forward as a promising approach to WSC, dramatic challenges are also recognized. First, the core of the dynamic service composition is the ability to identify, select, and integrate appropriate component services from a potentially large set of candidate Web services. That is, it requires an effective mechanism to find a composition of individual services that satisfies a variety of requirements. Second, another fundamental problem in service composition is to specify individual services at an appropriate level that enable precise understanding of



SDL

*lacks information for reasoning on not just what the inputs and outputs of a service are, but also what the inputs and outputs actually mean.*

services and allow on-demand composition. WSDL does not deal with the dynamic composition of existing services. Although WSDL defines a standard way for service description, it lacks information for reasoning on not just what the inputs and outputs of a service are, but also what the inputs and outputs actually mean. Therefore, when services developed by different organizations use different semantic models to describe services, a compiler that can translate a Web service description language (e.g., DAML-S or OWL-S) into an agreed standard service description language is mandatory. Third, a composition manager is required to control the invocation of individual atomic Web services and the data transfer between them.

There has been some research on dynamic Web service composition. In Rao et al.'s (2003) work, the description of existing Web services written in WSDL was translated into logical axioms in Linear Logic (LL) (Girard, 1987) and the requirements of composite services were specified in the form of an LL sequent to be proven. Then, it used a theorem prover to determine whether user requirements can be fulfilled by composition of atomic services. If the composition could address requirements, then the flow model(s) would be constructed. Sirin, Hendler, and Parsia (2003) developed a service composition prototype that had two basic components: a composer and an inference engine. The inference engine, which was an OWL (Web Ontology Language) reasoner, stored information about existing services in its knowledge base and was capable of finding matched services. The composer presented available choices to the user at each step. The service matching was done using the information in service profiles. The prototype executed the composition by invoking each individual service and passing the data between services according to the flow constructed by the service consumer.

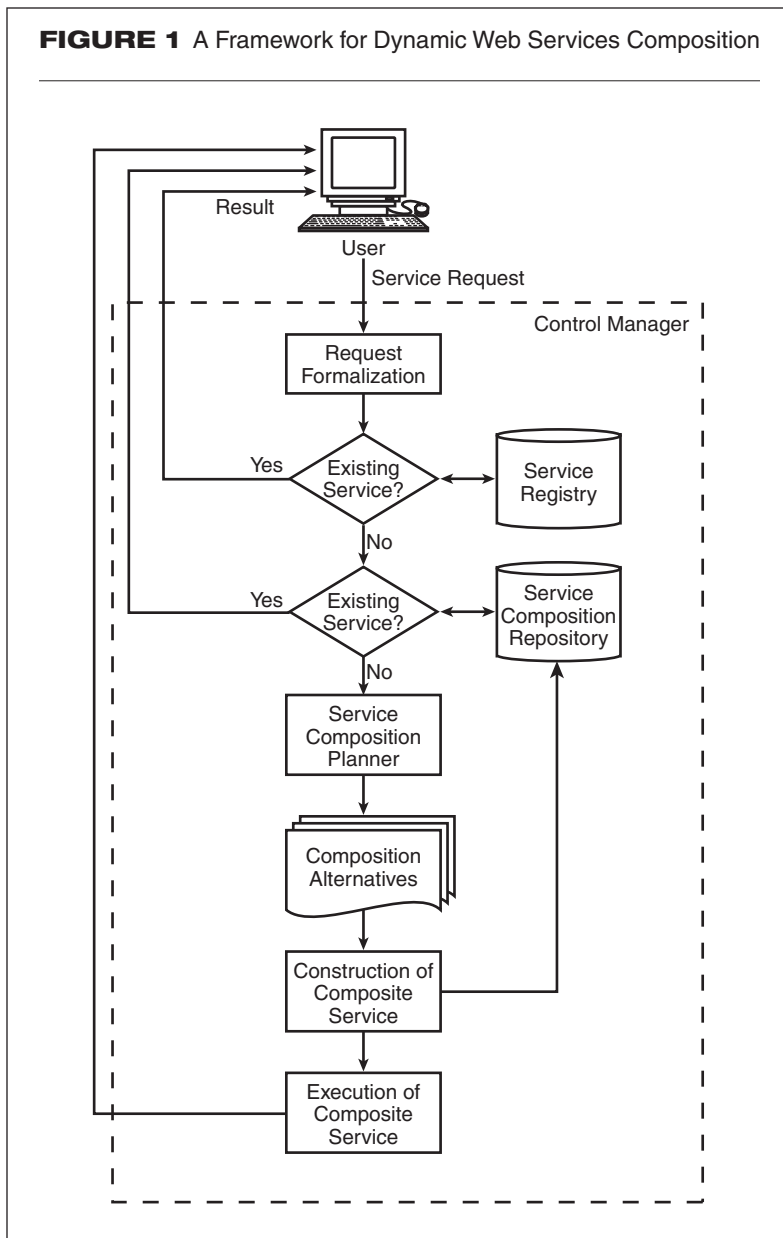
### **Personalization of Web Services**

Personalization is defined as any action that adapts information or services to the needs of a particular user or a group of users. Amazon.com, for example, automatically displays a customer's recent purchasing and viewing history when she logs in to the Web site. After the customer adds a product to her shopping cart, the Web site will display a list of other products that other customers have frequently purchased along with this product, assuming that those

products may also be of interest. In this case, personalization is enhanced by the collaborative effort.

Most existing Web services are not designed and implemented for personalized service consumption. In many situations, the service selection requires taking needs and preferences of individual consumers into consideration. For example, people may be interested in different types of news. A news alert Web service should be able to tailor daily news and select content that can specifically satisfy individual needs. Another typical example is target-advertising services, which automatically send electronic advertisements to potential customers based on their profiles. Apparently, understanding individual interests is crucial to ensuring that each advertisement is sent to customers who are likely to have interest in it.

Web personalization is currently known as a key to success in business today and in the future (Allen et al., 2001). A typical method for personalization is user-direct solicitation, such as *Yahoo! Personalize*. First, a static profile is created for each user at the very beginning by asking the user to specify his interests explicitly. In subsequent user-system interactions, user identity is detected and static profiles are used for automatic customization. The service personalization task is further confounded while taking potential changes of users' needs and preferences into consideration. In that case, it requires a mechanism to keep track of individual users' behavior and to update user profiles effectively when an interest shift is detected, making profiles reflect the real-time information or service needs of users. Another method is to use user feedback to update user profiles. To this end, there are two types of approaches: (1) explicit feedback and (2) implicit feedback. In the first approach, users are asked to provide explicit feedback about the relevance of provided information or service(s). The feedback is used to create or update user profiles. In the second approach, users do not have to provide specific feedback. Their interactions with the system (e.g., online browsing behavior) are captured and analyzed to infer their interests. Web-usage mining (WUM) is the process of discovering and interpreting patterns of user access to Web information systems by mining the data collected from the user interactions with the system. The user access patterns discovered through WUM are applied to identify the needs and preferences of each individual user, and subsequently to customize the content or structure of information or services.

**FIGURE 1** A Framework for Dynamic Web Services Composition

Information personalization has been recognized as one of the key tasks to cope with the information overload problem and satisfy the different needs of individual users. Similarly, given a large number of Web services and service providers, we believe that taking preferences of individual service consumers into consideration is essential to the process of dynamic service composition planning. It can improve the quality of services and consumers' satisfaction. In the next section we present an innovative framework for dynamic WSC that integrates a multi-criteria service selection policy and personal preferences while selecting atomic services for a service composition plan.

## A FRAMEWORK FOR DYNAMIC WEB SERVICE COMPOSITION

We propose a generic framework for dynamic and personalized Web service composition that highlights an integrated approach to the selection of component services. It takes three factors into consideration: (1) functional attributes of Web services, (2) nonfunctional attributes of services, and (3) consumer preferences. The framework is shown in Figure 1.

After a user's service request is received, it is converted into a logic format. Then, the control manager checks the service registry to see if any existing single Web service can fulfill the request. If yes, then the service is executed and results will be returned to the service consumer. Otherwise, the control manager examines a service composition repository, which stores previously created composite service plans and user requirements, to check if any of them can fulfill the request. If yes, the identified composite service is confirmed and executed, and results will be returned to the service consumer. Otherwise, the control manager will require the service composition planner to select qualified atomic services and make an appropriate composition plan. Once an optimal service composition plan is determined, it is passed on to the construction phase, where the preparation for composite service execution is performed. The generated composite service is stored in the composite service repository for possible future reuse. While planning offers the ability to automate the service composition process, finding a solution can be a problem of arbitrary complexity. Therefore, it is beneficial to reuse existing, verified compositions when possible. Finally, the generated composite service plan is executed.

### Identification of Candidate Web Services

Identification of potential component services is mainly based on functional attributes of services because every candidate component service should satisfy certain functional requirements specified by a service consumer. This is not a trivial process. A practical challenge is that Web services are very likely developed by different organizations, which may use different semantic models to describe services. It is necessary that service consumers and providers use a shared ontology to denote concepts while describing services so that the description of services can be understood appropriately (Rao et al., 2003).

**The functionality of a Web service should be described by a semantic annotation of what it does and by a functional annotation of how it works.**

The functionality of a Web service should be described by a semantic annotation of what it does and by a functional annotation of how it works. To cope with the heterogeneity of service description, some researchers have attempted to use ontology and Semantic Web technologies for WSC (Sirin et al., 2003). According to Gruber (1995), the term “ontology” means a *specification of a conceptualization*. In the context of Web services, an ontology is a description of individual Web services and their relationships that can exist for a community.

Ontology interoperability is important in service composition because semantic annotations of Web services have been widely discussed in the Semantic Web community. The Semantic Web (Berners-Lee et al., 2001) relies on ontologies to formalize domain concepts shared among services. Its aim is to enable greater access not only to content, but also to services on the Web. Service consumers should be able to discover, invoke, compose, and monitor Web resources offering specific services and having particular properties in an automated manner. DAML-S (Ankolekar et al., 2001) is a Web service ontology that provides a core set of markup language constructs for describing the properties and capabilities of Web services. It uses the *Service* class to model Web services with the properties *presents*, *describedBy*, and *supports*. The properties in turn have classes *ServiceProfile*, *ServiceModel*, and *ServiceGrounding* as their respective ranges. The *ServiceProfile* gives a high-level description of a service that contains inputs, outputs, preconditions, and post-conditions of the service. It has similar functionality to the yellow pages in UDDI, and can be used by clients to select and locate services from registries. The *ServiceModel* is a detailed description of a service in which it is modeled as a process. It is similar to the business process model in BPEL4WS. The *ServiceGrounding* provides the binding level information of how a client can access a service. It is a mapping from DAML-S to WSDL. An example is the DAML-S Matchmaker (Paolucci et al., 2002) which was a system that augmented the standard UDDI architecture with semantic service description. It allowed location of services based on their capabilities to support WSC.

Recently, DAML-S has been extended to OWL-S (Web Ontology Language for Services). OWL is a revision of the DAML+OIL Web ontology language. It is “intended to be used when the information contained in documents needs to be processed by applications, as opposed to

situations where the content only needs to be presented to humans. OWL can be used to explicitly represent the meaning of terms in vocabularies and the relationships between those terms...” (W3C, 2004). The OWL-S specification represents the first W3C member submission in the area of Semantic Web Services, where Semantic Web technology is applied to the challenges offered in the Web services field. OWL-S is more expressive to represent meaning and semantics than XML, RDF, and RDF-S. It goes beyond those languages in its ability to represent machine-interpretable content on the Web. OWL-S supplies a core set of markup language constructs for describing properties and capabilities of Web services in an unambiguous, computer-interpretable form. It is comprised of three main OWL ontologies (namely, Profile, Process, and Grounding), a Service Ontology, and four additional ontologies serving supplementary roles (i.e., Logical Expression Constructs, List Constructs, Profile Additional Parameters, and Actor Ontology). OWL-S facilitates automation of Web service tasks, such as Web service discovery, execution, composition, and interoperation (W3C, 2004).

We propose to use OWL-S to describe the properties and capabilities of Web services. A logic converter can transform the service description into first-order logic automatically, in which each sentence or statement is broken down into a subject and a predicate. The predicate modifies or defines the properties of the subject. As a result, a service can be represented by a rule that expresses the output of the service given certain inputs. The SWORD system (Ponnekanti and Fox, 2002) used similar rule-based representation of Web services.

### **Selection of Atomic Services for Composition**

Service composition emphasizes the global optimum by aggregating individual services through planning. A set of selected Web services must satisfy additional requirements in order to be composed into a complex service. Such requirements include connectivity, correctness, and scalability (Milanovic and Malek, 2004). For example, every composition approach must guarantee connectivity, which indicates which services are composed and what will be input and output messages passed between ports; correctness of composition requires that properties of a composed service must be verified; and scalability demands that composition frameworks scale with the number

***In our framework, we propose a three-phase process for selecting atomic services and making service composition plans that can best satisfy consumers' requirements.***

of composed services, because it is likely that a complex service will involve many existing services in an invocation chain (Milanovic and Malek, 2004)

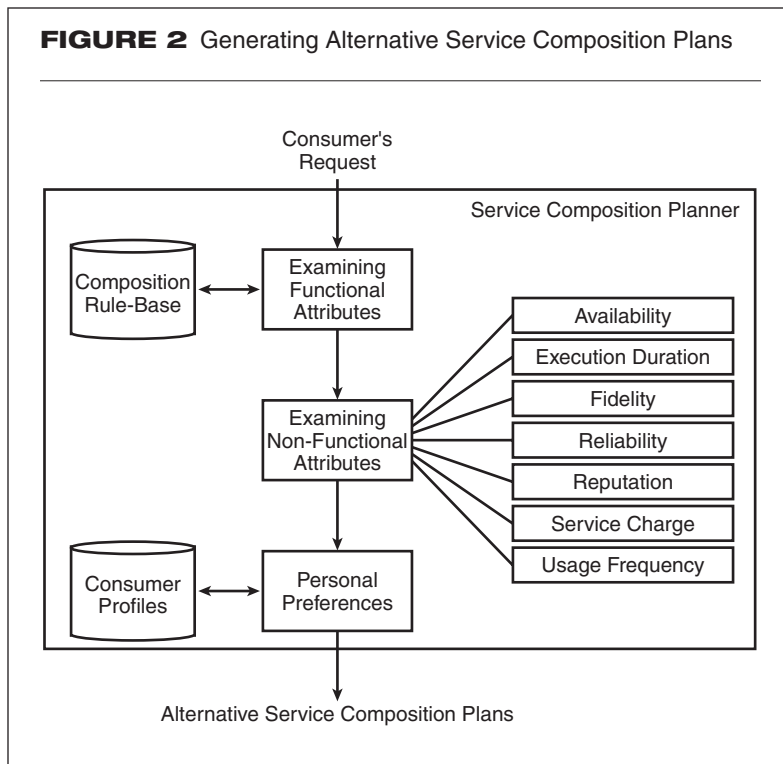
Traditionally, a service-selection component filters candidate services only based on the match between the consumer's functional goals and the functional attributes of services. WSC rests upon the coordination and collaboration among multiple Web services. Under-par performance or failure of any single Web service may result in the failure of entire composed services. The majority of current approaches are based on an assumption that individual Web services are always available and well behaved. In reality, however, this is not always true. While selecting services to generate an optimal service composition plan in order to satisfy a service request, it is essential to develop and use a set of criteria for evaluating individual services. The evaluation criteria should be based on both objective and subjective features of Web services. The former is mostly function oriented, which verifies whether the requested functionality can be fulfilled by a service. The latter is nonfunction oriented (e.g., quality of service, perception, reputation, service charge, trust, response time, etc.), in which services are judged on the basis of service quality and human perceptions. The objective measures are more fundamental, while subjective measures become increasingly important, especially when objective criteria are met or when there are too many options. For example, Zeng et al. (2003) developed a multidimensional measure of quality of service for WSC. After examining the functional and nonfunctional attributes of services, the composition planner should also take individual preferences (e.g., preferred product brand(s) and service providers) of service consumers into consideration.

Some recent research on Web service composition uses AI planning techniques to deal with the dynamics of the environment and services (Pistore et al., 2004; Madhusudan and Uttamsingh, 2004; Carman et al., 2003; Srivastava and Koehler, 2004). The desired output of a planning problem is a sequence of actions that can be executed in order to achieve the desired goal. Madhusudan and Uttamsingh, 2004) proposed an integrated service planning and execution architecture for supporting service composition under dynamic conditions based on domain independent AI planning and machine learning techniques. In their approach, service requests are received from consumers as a de-

clarative goal-driven language. After alternative service plans are generated, an optimal plan is selected and then executed. Changes in the execution environment are monitored and may trigger replanning to generate a new service plan. Many areas of planning may contribute to service composition, such as distributed planning (DesJardins et al., 1999) and planning as model checking (Giunchiglia and Traverso, 1999). For example, the approach to planning as model checking can provide initial solutions to the generation of complex plans. Pistore et al. (2004) proposed a "Planning as Model Checking" framework for planning under uncertainty, which allowed nondeterminism in the initial state and in the outcome of action execution. A plan is modeled as a system with an internal dynamic, which monitors the evolution of the domain via observations describing the visible part of the domain state and controls the evolution of the domain by triggering actions. The Hierarchical Task Networks (HTN) planning approach (Erol et al., 1994) decomposes a task into a set of sub-tasks, which are further divided into another set of sub-tasks, until the resulting set of tasks only consists of atomic tasks (i.e., services) that can be executed directly. These planning techniques map a problem to a classical representation involving initial states and goals.

In our framework, we propose a three-phase process for selecting atomic services and making service composition plans that can best satisfy consumers' requirements (see Figure 2). The first phase is to examine the functional attributes of services, during which an inference engine uses logic representation and a backward chaining search to produce a set of chains of services that can produce desired outputs, starting from the goal to the given inputs. User requirements in terms of functionality of the requested service and related constraints (e.g., the budget and delivery date) are considered. Such a rule-based approach to service composition has been used to support parameterization, dynamic binding, and flexible service compositions (Orriens et al., 2003).

Although different Web services may provide similar functionalities, their nonfunctional property values may be different. Therefore, in the next stage, nonfunctional properties are employed to select services. For each atomic service  $i$  in a service plan  $j$  that consists of a chain of  $N$  services  $S_j = \{s_{j1}, s_{j2}, \dots, s_{jN}\}$ ,  $Q(s_{ji})$  ( $1 \leq i \leq N$ ) is the estimated quality of the  $i$ th service. We consider seven generic nonfunctional attributes based on (Zeng et al., 2003):

**FIGURE 2** Generating Alternative Service Composition Plans

1. *Availability (q1)*: the availability of a service  $s_i$  refers to the probability that  $s_i$  is accessible. It can be computed as the ratio of the number of successful accesses to the total number of accesses based on historical data.
2. *Execution duration (q2)*: the time delay between the time when a service request is issued and the time when results are received. It includes process time and transmission delay.
3. *Fidelity (q3)*: the accuracy of services' operations. It can be measured by the accuracy of results of prior service executions.
4. *Reliability (q4)*: it corresponds to the likelihood that an atomic service will be performed when the consumer demands it, and it is a function of failure rate (i.e., the probability of a service to be executed). It can be computed by the percentage of times when a service consumer receives a response from the service within the time limit.
5. *Reputation (q5)*: the reputation of a service can be obtained either from service rankings in the service community, or can be computed as an average of consumers' perceived satisfaction levels with the service.
6. *Service charge (q6)*: the amount of associated charges that a user has to pay for consuming a service.

7. *Usage frequency (q7)*: the number of times that a service has been consumed by users within a fixed time period.

The values of the above nonfunctional attributes can be derived from historical service records stored in service logs or customer ratings (e.g., product ratings from customers on E-business Web sites such as Amazon.com). After computing the values of the above nonfunctional attributes for each atomic service, in the third phase, the service composition planner will check the consumer preferences stored in the consumer profiles, if applicable. Consumer preferences may include importance weights assigned to the quality attributes by individual consumers, or even service providers or product brand(s) that customers prefer. For example, among the above quality of service criteria, one service consumer may consider  $q_1$  and  $q_4$  as much more important than  $q_5$  and  $q_6$ , but another consumer may consider the service charge ( $q_6$ ) the most important. Allowing consumers to specify their preferences can help the service composition planner select an optimal composition plan. Individual preferences can be derived from either direct user solicitation or consumer profiles. Finally, the overall estimated quality of the  $i$ th service in the  $j$ th service plan can be obtained as

$$Q(s_{ji}) = \sum_{k=1}^7 w_k * q_k(s_{ji}) \quad (1)$$

where  $w_k$  is the importance weight of the  $k$ th criterion, and  $q_k(s_{ji})$  is the normalized value of the  $k$ th criterion. Note that there are other methods to compute the overall quality of a service. The final result after this three-stage process can be a set of alternative plans ranked in a decreasing order of their estimated quality of service to the service request. The optimal plan will be selected and executed.

## A PROTOTYPE SYSTEM

### Real Estate E-Business Supported by Web Services

A single real estate transaction is intricate because it consists of several processes, involves a number of people (e.g., buyer, seller, real estate agents, notaries, and home inspectors) and service providers (e.g., mortgage companies, title companies, and home insurance companies), and requires coordination and management of these people and processes to serve the best interest of buyers and sellers. Web services can

**TABLE 1** A List of Web Services for Home Buying

Services	Description
Listing of real estate agents	Generate a list of real estate agents based on buyer criteria, such as region, company, and years of experience
Home directory service	List all houses that are available on the market
Home listing alert service	Deliver a list of houses that satisfies the buyer criteria, such as the house type, location, age, price range, number of bedrooms, etc. (a personalized service)
Mortgage service	Preapproving loan application and final loan approval
Contract service	Provide the service of writing and submitting a contract
Home inspection service	Provide services of house inspection
Home insurance service	Provide home insurance service for a particular house
Criminal record service	Check the criminal record of a specific neighborhood
School information service	Provide the names and rankings of the elementary, middle, and high schools in the region where a specific house is located
Title service	Provide the title search and title insurance services for a specific house
Settlement service	Provide services related to the closing of a house-buying transaction
Utility service	A utility company (e.g., electricity, phone) that provides services to a house

revolutionize the real estate industry by streamlining these interactions.

Buying a house is a process involving many interactions and transactions (e.g., making an offer, applying for a loan application, and conducting a title search) among multiple business partners (e.g., buyer, seller, buyer agent, seller agent, mortgage firm, home inspection company, title search and insurance company). It consists of activities that have standard procedures to follow and standard documents to use. Some activities also involve dynamic factors such as constantly changing mortgage rates, the availability of houses on the market, and the availability of multiple providers for services such as title search and mortgage service. Therefore, there are opportunities for automating these activities and facilitating the workflow by using Web services.

Home buyers are usually interested in getting home listings, reading related articles on the trend of real estate markets and tips on how to choose an agent and buy a house, finding reliable home inspectors and mortgage service providers, etc. After finding a house, there is still a lot of routine paperwork involved, such as the management of finances, taxes, and insurance. Obviously, buyers need help in going through such a complicated process to avoid possible problems. Web services technology allows different people and service providers to interact and achieve a more efficient and effective way of buying or selling houses. Some possible services that the real estate community can implement as Web services for home buyers are listed in Table 1. Some of them can be carried out in parallel, such as criminal record service

and school information service, while other services may have to be carried out in a specific order. For example, the contract service must be consumed before the home inspection service, which is usually consumed before title and home insurance services.

Relationships among some services can be described in the form of production rules as follows:

- Home directory service, criminal record service, school information service → Home listing alert service
- Listing of real estate agents, mortgage service → Contract service
- Home insurance service, title service, mortgage service → Settlement service

Taking a personalized home listing alert service as a simple example, in the first step, based on backward chaining in rule-based reasoning, the planner finds that in order to consume the home listing alert service, a criminal record service, a school information service, and a home directory service must be available and composed. The second step is to select alternative atomic services that provide those three types of services. A criminal record service is special in that county or city police departments are the authorities in providing information about the criminal history records. A school information service usually involves retrieval of officially published test scores from local and national departments of education. The most frequently used home directory service would be multiple listing services (MLS). Other sources may include *for sale by owner* (FSBO) service or electronic copies of the real estate section in local

newspapers. MLS is widely used for its availability, reputation, usage frequency, fidelity, and reliability. The outcome of this step is a list of different combinations of the above services (i.e., a list of alternative plans). The third step is to rank those combinations using the formula in Equation (1) based on quality of services and user preferences (e.g., assigned weights). A service composition plan that has the highest overall quality is then identified. Finally, the control manager personalizes the home directory service based on individual buyers' preferences stored in the profiles of buyers. For example, a buyer is interested in four-bedroom, single-family houses that were built within the past 15 years, located within 15 miles of his working place and within a good school district, with a two-car garage, and no less than 2200 square feet of living space. The generated home listing alert service will automatically deliver new house information that matches the above criteria. In the meantime, it also keeps track of the house information that the buyer actually views to determine his true interest. The buyer can also rate each house as he views it. Such information can be used by the system to detect potential changes in the buyer's interest and adjust the corresponding profiles for future service personalization.

By wrapping existing business services as Web services and applying the proposed framework for service composition, a real estate business can extend the life cycle of existing applications and enhance interaction with other associated companies, thus enabling the business to close a deal faster and gain competitive advantage.

### **A Prototype System for Home Purchasing**

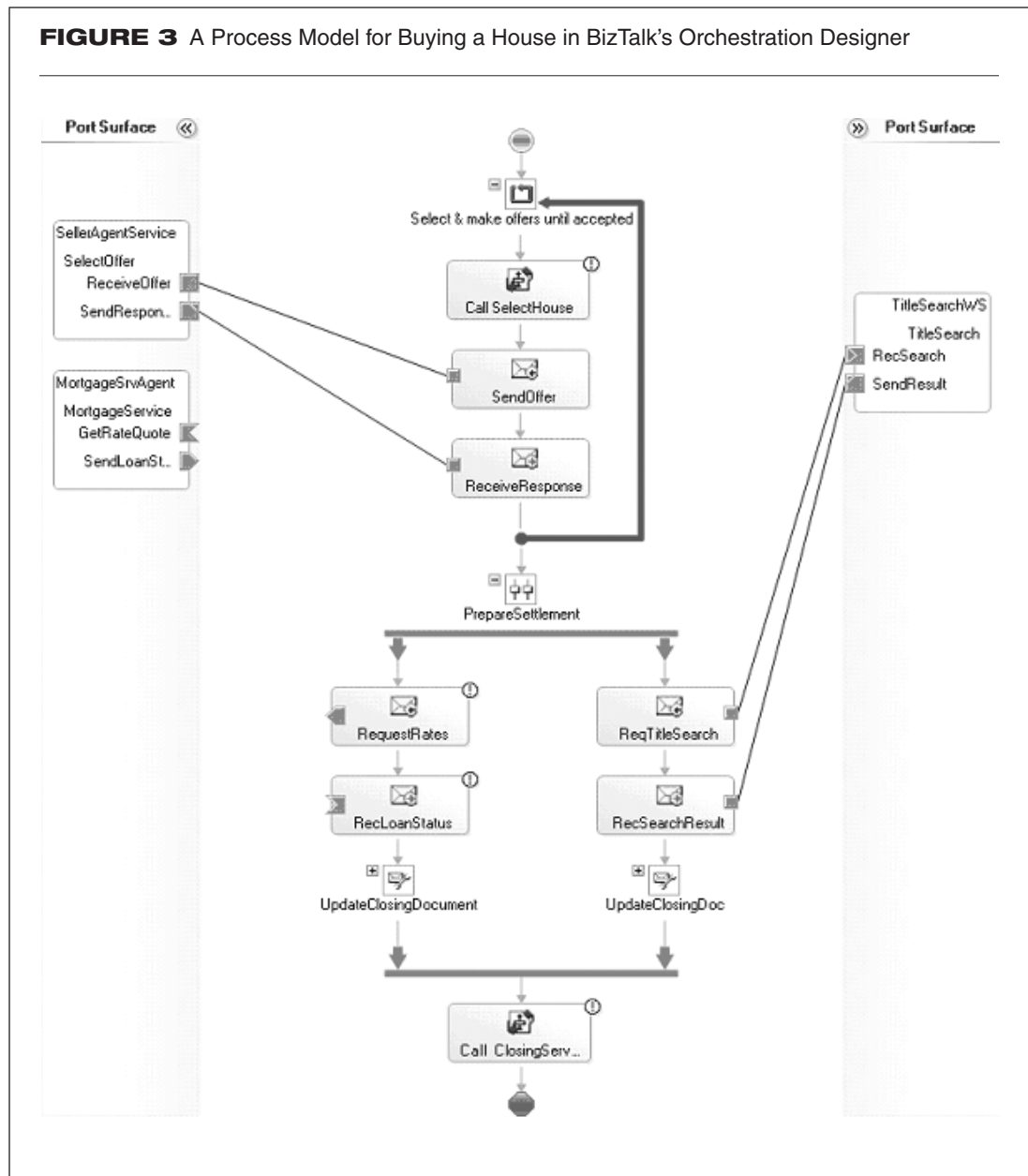
In this section we use a simplified scenario of buying a house to exemplify the dynamic and personalized Web services composition framework presented in the previous section. We chose a commercial business process management and integrated software product, Microsoft BizTalk 2004, as a main tool to develop our prototype system. Visual Studio.net was used to develop a customized Web service selection and composition agent that used a private Web service registry to support dynamic invocation of Web services. BizTalk 2004 supports BPEL, an industry standard for Web services composition. ASP.NET was also used to develop Web-based user interfaces for requesting and consuming the underlying Web services. The

prototype system presented here serves as a proof of concept for the proposed framework.

**A Process Model of BuyHouse.** According to the precedence constraints defined in the production rules, we constructed a home-buying process called *BuyHouse* using the Orchestration Designer in BizTalk 2004, as shown in Figure 3. The process was modeled approximately from a viewpoint of designing a real estate Web portal based on activities involved in a home-buying cycle. It is a simplified model so that we can illustrate the essential concepts more clearly.

This process involves the following sequence of steps:

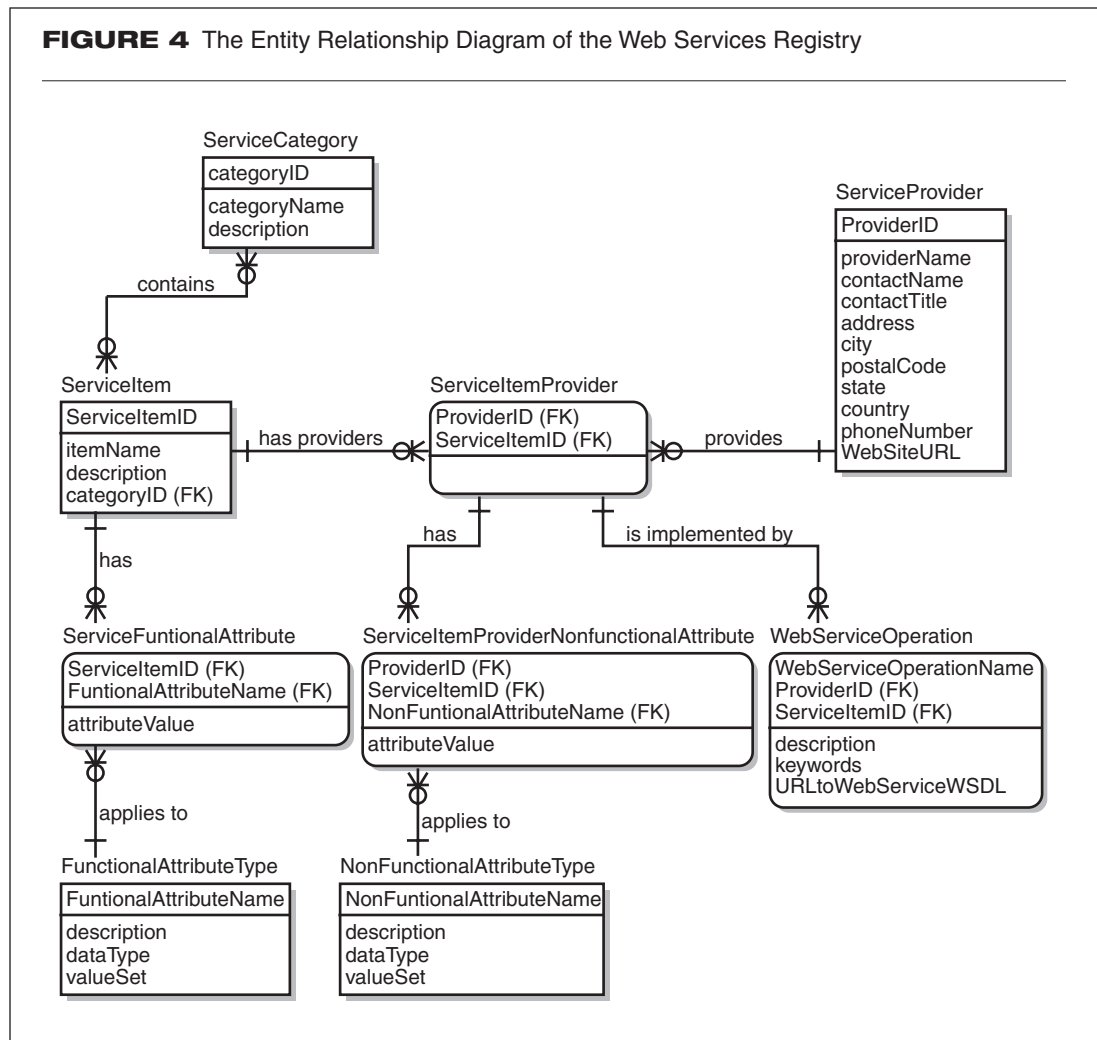
1. *Call SelectHouse.* Because the house selection sub-process is complex and involves a lot of human interactions, it is carved out as a separate Human Workflow Service (HWS) orchestration named *SelectHouse* that is called from the *BuyHouse* process. This is an example of a nested process (i.e., process decomposition) in BizTalk. The primary orchestration calls the secondary orchestration *SelectHouse* to accomplish the house selection activity. Information about a house that the buyer may be interested in making an offer or will be returned. This secondary orchestration involves a couple of activities that require intensive human interactions:
  - a. The buyer agent needs to elicit the buyer's requirements, including price range, preferred areas to live, expected settlement time, school district preference, preferred builders, etc.
  - b. An intelligent agent can automatically select a list of properties that matches the buyer's requirements and notify him by e-mail or other means to view the detailed information. A physical visit to an interested house before making an offer is quite common.
2. *SendOffer.* After a buyer is ready to make an offer on a selected house, a *SendOffer* service is invoked to send an electronic offer to the *SellerAgentService*. An offer document contains offered price, finance, expected closing date, and other contingencies.
3. *ReceiveResponse.* After receiving and assessing an offer, the *SellerAgentService* returns a response that may accept the offer, reject it, or provide a counter-offer. If the offer is accepted, the process will enter

**FIGURE 3** A Process Model for Buying a House in BizTalk's Orchestration Designer

the PrepareSettlement phase; otherwise, it will go back to Step 1.

4. *PrepareSettlement*. Once an offer on a house is accepted, several activities can occur in parallel to prepare for the final settlement. These activities include obtaining a home mortgage, conducting a title search, performing a home inspection (optional), selecting a settlement firm for closing, getting home insurance, etc. To keep it simple, we do not show all of these activities in the BuyHouse orchestration model in Figure 3. The system keeps track of settlement activities and will update the records to reflect the status of the preparation. The primary orchestration can store updated settlement status messages in a folder for the buyer or
- his agent to track the status of these activities.
5. *Call ClosingService*. Once all the parallel activities included in the PrepareSettlement have been finished, a ClosingService orchestration will be called to complete the BuyHouse process.

A BizTalk Server orchestration can also be exposed as a Web service and consumed by a Web service client such as a Web application implemented in ASP.NET. An ASP.NET application can construct an object based on the Web reference to the orchestration and use it to interact with a remote Web service. It populates data in this object and submits the object as a parameter in a Web service request to the

**FIGURE 4** The Entity Relationship Diagram of the Web Services Registry

orchestration. The orchestration processes the submitted data and returns a corresponding message to the ASP.NET application.

**A Private Web Service Registry Supporting Dynamic Web Service Invocation.** Public UDDI registries are too generic and lack control over the quality of Web services and the trustworthiness of service providers. Searching for candidate Web services that satisfy a consumer's request from such registries can result in many useless entries. Therefore, in this prototype system, instead of using a public UDDI registry, we created and used a domain-specific private Web service registry (WSR) and a set of Web service APIs to the WSR to enable the business process to retrieve appropriate Web service(s) dynamically. The data model of the Web service registry depicted in an entity relationship diagram is shown in Figure 4.

The entity type *ServiceCategory* was used to define high-level categories for individual

service items. For example, "mortgage" is an instance of *ServiceCategory*. Service items belonging to a single category can be identified first. Then, specific service items can be selected based on values of their functional attributes stored in the *ServiceItemFunctionalAttribute* entity type. Metadata of functional attributes was stored in the *FunctionalAttributeType* entity type. For example, based on an instance of *ServiceCategory* "mortgage," we can find service items (e.g., mortgage products/packages) offered by different mortgage firms, which include varieties of mortgage types such as fixed-rate mortgage, ARM (adjustable rate mortgage), or interest-only mortgage. Basic information about all service providers was stored in the *ServiceProvider* so the system could identify all qualified providers for a requested service item via the associative entity type *ServiceItemProvider*.

Web service operations provided by service providers were stored in the *WebServiceOperation* entity type that has attributes such as description, keywords, and *URLtoWebService*

**I**n practical implementations, several Web service operations can be implemented in a single Web service.

The `URLtoWebServiceWSDL` attribute contained the URL pointing to the WSDL file of a selected Web service that implements specific Web service operation(s) by a service provider (identified by `ProviderID`).

Once a list of service providers that offer the same service item is found according to their functional attributes, corresponding URLs of WSDL files of the Web service operation implemented by each provider can be retrieved from the WSR based on the relationship between `ServiceItemProvider` and `WebServiceOperation`. We can then call these Web service operations dynamically to retrieve up-to-date information from those providers and select the best service provider based on values of nonfunctional attributes such as service charge, availability, and service delivery time (i.e., execution duration). The attribute `PriceValidDateTime` can be used to determine whether pricing information needs to be updated by requesting a price quoting service from providers; otherwise, the cached values of nonfunctional attributes stored in the `ServiceItemProviderNonFunctionalAttribute` entity type can be used to improve the performance of the system.

We have developed a Web service API to provide interfaces to the Web Service Registry. Some commonly used operations defined in the Web service interfaces to WSR and their functionalities are listed as follows:

- *GetProviders (FunctionalAttributeCriteria)*: a set of providers' IDs will be returned based on a set of criteria specified by the consumer and values of functional attributes associated with a service.
- *GetProviderWSDL (ServiceItemID, WebOperationName)*: returns a list of URLs of WSDL files from service providers that implement a specific Web service operation (the name of the operation is specified by the second parameter).
- *SearchServiceItems (FunctionalAttribute Value Set)*: names and values of a set of attributes will be used as parameters to find out appropriate service items. A number of service item IDs will be returned as the result of this Web operation.

**Dynamic Invocation and Composition of Web Services.** Figure 5 depicts a high-level architecture about how BuyHouse orchestration, SelectHouse orchestration, Mortgages

Service Agent (MSA), Web Service Registry, and a real estate portal interact during dynamic Web services invocation and composition. The portal is an interface with which a home buyer can interact with Web services exposed by orchestrations and MSA. The major steps carried out by the MSA, as labeled in Figure 5, are described in Table 2.

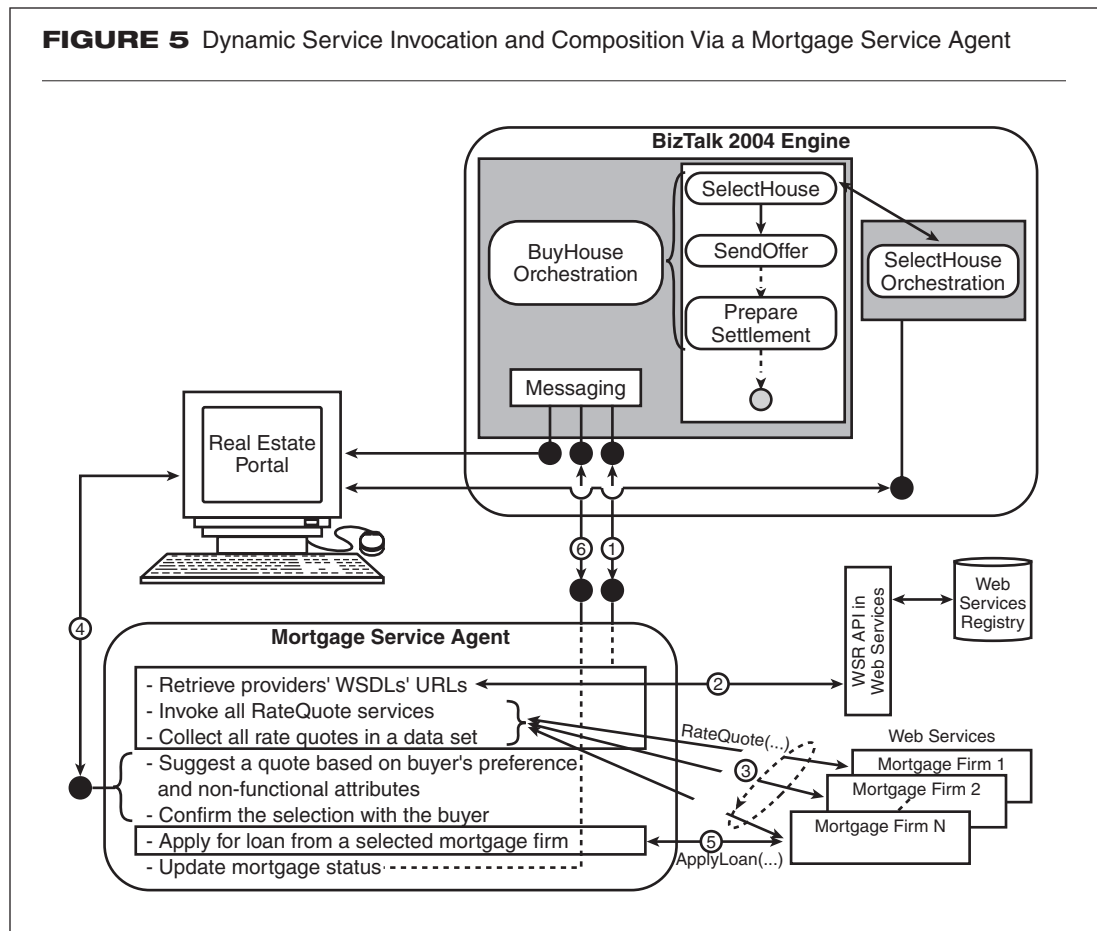
Similar steps can also be carried out simultaneously for selecting a title company and a home insurance company. In general, functional attributes stored in a WSR are often used to identify all candidate providers for a requested service item. Personal preferences and non-functional attributes are used to determine which providers' Web services should be selected for composition. Values of nonfunctional attributes often change over time; therefore, invoking Web operations to query qualified providers to collect these attribute values in real-time is necessary.

Domain-specific standards for Web services are essential for the success of dynamic Web service selection, composition, and invocation. As mentioned earlier, providers that offer the same service items should follow predefined standards while developing interfaces of Web services to realize interoperability. There are increasing efforts in the development of XML-based document standards, but Web service operation standards for developing applications in specific domains remain scarce.

## CONCLUSION

Dynamic Web service composition is emerging as a new way of empowering E-businesses. Building composite Web services can save extensive time and cost for developing new applications and enhance the interoperability and collaboration among E-business partners. Seamless composition of Web services has enormous potential in streamlining business-to-business processes and the integration of enterprise applications. The applications can determine not only what existing services can be bound together, but also how they should be carried out at runtime in order to fulfill users' complicated requests. However, WSC is a very challenging task due to the potentially large number of services that provide the same functionality, as well as the heterogeneity and changing nature of services.

This article has identified several important issues involved in Web service composition and presented a framework for dynamic Web service composition that adopts an innovative

**FIGURE 5** Dynamic Service Invocation and Composition Via a Mortgage Service Agent

approach to selecting and composing services based on functional and nonfunctional attributes and individual preferences. The framework enables finding an optimal composition of services

at runtime. We have also demonstrated an application of the framework using an example in the real estate domain and discussed some implementation details. ▲

**TABLE 2** Major Steps Carried Out by the Mortgage Service Agent (see [Figure 5](#))

1. Invoke the MSA's GetRateQuotes Web operation by carrying out the RequestRates activity in the BuyHouse orchestration. The code for the GetRateQuotes operation was written in Visual Basic .NET as follows:

```

1. <WebMethod()> _
2. Public Function GetRateQuotes(ByVal ServiceItemID as String,
   ByVal NumOfYear As Integer, ByVal Amount As Integer, ByVal
   myProfile As CustomerProfile) As DsRateQuoteResult
3. Dim MyMortgageRegistry As New MortgageRegistry.MortgageQuery
4. Dim MyRateQuoteResult1 As New DsRateQuoteResult
5. Dim DsRateQuoteWSDL1 As New MortgageRegistry.DsRateQuoteWSDL
6. Dim MortgageWSDL As MortgageRegistry.DsRateQuoteWSDL.
   RateQuoteWSDLRow
7. Dim MortgageFirmWSDL As String
8. Dim MyRateQuoteWS As RateQuoteWS.WSQuery
9. Dim MyRateQuoteDs As RateQuoteWS.DsRateQuote
10. Dim drQuoteResult As DsRateQuoteResult.RateQuoteResultRow
11. Dim dtQuoteResult As New
   DsRateQuoteResult.RateQuoteResultDataTable
12. DsRateQuoteWSDL1 = MyMortgageRegistry.GetProviderWSDLs
   (ServiceItemID, "RateQuote")
13. For Each MortgageWSDL In DsRateQuoteWSDL1.Tables(0).Rows
14. MyRateQuoteWS = New RateQuoteWS.WSQuery(MortgageWSDL.WSDL)
15. MyRateQuoteDs = MyRateQuoteWS.RateQuote(MortgageType,
   NumOfYear, Amount, myProfile)
16. Dim MyRateQuoteDt As
   RateQuoteWS.DsRateQuote.RateQuoteDataTable
17. MyRateQuoteDt = MyRateQuoteDs.Tables(0)
18. Dim MyRateQuoteDr As RateQuoteWS.DsRateQuote.RateQuoteRow
19. If Not (MyRateQuoteDt Is Nothing) Then
20. drQuoteResult = MyRateQuoteResult1.RateQuoteResult.
   NewRateQuoteResultRow
21. MyRateQuoteDr = MyRateQuoteDt.Rows(0)
22. drQuoteResult.MortgageFirmID = MortgageWSDL.MortgageFirmID
23. drQuoteResult.Rate = MyRateQuoteDr.Rate
24. drQuoteResult.ApprovalTime = MyRateQuoteDr.ApprovalTime
25. MyRateQuoteResult1.RateQuoteResult.Rows.Add(drQuoteResult)
26. End If
27. Next
28. Return MyRateQuoteResult1
29. End Function

```

2. Retrieve URLs of candidate service providers' WSDL files: mortgage lenders that offer the requested mortgage package such as "30-year, fixed rate" will be identified. The URLs of the WSDL files for individual Web services that implement the RateQuote( ) operation are retrieved from the WSR (line 12 of the above code).
3. Invoke all RateQuote services dynamically and collect all mortgage rates. Each candidate mortgage lender selected from the previous step will be included in a loop. Inside the loop, the Web service of each mortgage firm that provides the requested mortgage service will be invoked and each returns a mortgage rate quotation. This step was implemented with code from line 13 to line 27.
4. Select a mortgage service: the system will assess functional attributes such as mortgage rate that is retrieved dynamically and nonfunctional attributes (e.g., reputation and reliability of a mortgage company). Applying Equation (1) for each candidate mortgage service, as well as considering the buyer's preference stored in a profile, the system will automatically select an optimal mortgage firm's service based on its overall quality and recommend it to the buyer (it still needs the buyer's approval). After the buyer accepts the recommendation, the system will initiate a loan application implemented within MSA.
5. The loan application will invoke the ApplyLoan( ) method of the mortgage Web service provided by the selected mortgage firm to start the loan application process.
6. The mortgage application status within the BuyHouse orchestration will be updated by calling an exposed Web operation RecLoanStatus( ).

## References

- Allen, C., D. Kania, and B. Yaeckel, *One-to-One Web Marketing: Build a Relationship Marketing Strategy One Customer at a Time*, 2nd ed., New York: John Wiley & Sons, 2001.
- Ankolekar, A., M. Burstein, J.R. Hobbs, O. Lassila, D.L. Martin, A.S. McIlraith, S. Narayanan, M. Paolucci, T. Payne, K. Sycara, and H. Zeng, DAML-S: Semantic Markup for Web Services, *International Semantic Web Working Symposium (SWWS)*, Stanford University, 2001.
- Berners-Lee, T., J. Hendler, and O. Lassila, The Semantic Web, *Scientific American*, (May 2001), 35–43.
- Carman, C., L. Serafin, and T. Paolo, Web Services Composition as Planning, *Workshop on Planning for Web Services*, Trento, Italy, 2003.
- DesJardins, M., E. Durfee, C. Ortiz, and M. Wolverton, A Survey of Research in Distributed, Continual Planning, *AI Magazine*, 20 (1999), 13–22.
- Erol, K., J. Hendler, and D.S. Nau, Semantics for Hierarchical Task Network Planning, Technical report, University of Maryland–College Park, 1994.
- Girard, J.-Y., Linear Logic, *Theoretical Computer Science*, 50 (1987), 1–102.
- Giunchiglia, F. and P. Traverso, Planning as Model Checking, *5th European Conference on Planning*, Durham, K, 1999.
- Gruber, T.R., Toward Principles for the Design of Ontologies Used for Knowledge Sharing, *International Journal of Human-Computer Studies*, 43 (1995), 907–928.
- Madhusudan, T. and N. Uttamsingh, A Declarative Approach to Composing Web Services in Dynamic Environments, *Decision Support Systems*, forthcoming.
- Martin, D., M. Paolucci, S. McIlraith, M. Burstein, D. McDermott, D. McGuinness, B. Parsia, T. Payne, M. Sabou, M. Solanki, N. Srinivasan, and K. Sycara, Bringing Semantics to Web Services: The OWL-S Approach, *First International Workshop on Semantic Web Services and Web Process Composition (SWSWPC '04)*, San Diego, California, 2004.
- Milanovic, N. and M. Malek, Current Solutions for Web Service Composition, *IEEE Internet Computing*, 8 (2004), 51–59.
- Orriens, B., J. Yang, and M.P. Papazoglou, A Framework for Business Rule Driven Web Service Composition, *International Conference on Conceptual Modeling (ER)*, Chicago, IL, Oct. 13–16, 2003.
- Paolucci, M., T. Kawmura, T. Payne, and K. Sycara, Semantic Matching of Web Services Capabilities, *First International Semantic Web Conference*, Sardinia, Italy, 2002.
- Pistore, M., F. Barbon, P. Bertoli, D. Shaparau, and P. Traverso, Planning and Monitoring Web Service Composition, *Workshop on Planning and Scheduling for Web and Grid Services in Conjunction with ICAPS*, Whistler, British Columbia, Canada, 2004.
- Ponnekanti, S.R. and A. Fox, SWORD: A Developer Toolkit for Web Services Composition, *The Eleventh International World Wide Web Conference*, Honolulu, Hawaii, 2002.
- Rao, J., P. Küngas, and M. Matskin, Application of Linear Logic to Web Service Composition, *International Conference on Web Services (ICWS '03)*, Las Vegas, NV, 2003.
- Sirin, E., J. Hendler, and B. Parsia, Semi-automatic Composition of Web Services using Semantic Descriptions, *Web Services: Modeling, Architecture and Infrastructure Workshop* in conjunction with *ICEIS '03*, Angers, France, 2003.
- Srivastava, B. and J. Koehler, Planning with Workflows — An Emerging Paradigm for Web Service Composition, *Workshop on Planning and Scheduling for Web and Grid Services ICAPS*, Whistler, British Columbia, Canada, 2004.
- W3C, OWL-S: Semantic Markup for Web Services, 2004, <http://www.w3.org/Submission/2004/SUBM-OWLS-20041122>.
- Zeng, L., B. Benatallah, M. Dumas, J. Kalagnanam, and Q.Z. Sheng, Quality Driven Web Services Composition, *World Wide Web*, Budapest, Hungary, 2003.
- Zhang, D., Web Services Composition for Process Management in E-Business, *Journal of Computer Information Systems*, 45: 2 (2005), 83–91.
- Zhou, L. and D. Zhang, Trust-Based Automatic Web Service Composition, *The Third Workshop on E-Business (Web 2004)*, Washington, D.C., 2004.