Service-Centric Framework for a Digital Government Application

Athman Bouguettaya, *Fellow*, *IEEE*, Qi Yu, *Member*, *IEEE*,
Xumin Liu, *Member*, *IEEE*, and Zaki Malik, *Member*, *IEEE*

Abstract—This paper presents a service-oriented digital government infrastructure focused on efficiently providing customized services to senior citizens. We designed and developed a Web Service Management System (WSMS), called WebSenior, which provides a service-centric framework to deliver government services to senior citizens. The proposed WSMS manages the entire life cycle of third-party web services. These act as proxies for real government services. Due to the specific requirements of our digital government application, we focus on the following key components of WebSenior: service composition, service optimization, and service privacy preservation. These components form the nucleus that achieves seamless cooperation among government agencies to provide prompt and customized services to senior citizens.

Index Terms—Web service management system, WebSenior, digital government, service composition, service optimization, privacy preservation.

1 Introduction

Service-oriented Computing (SOC) is a new and novel paradigm that holds real promise to move the field of computing from a data-centric to a service-centric view [20]. The simple and effective Service-Oriented Architecture (SOA) has contributed greatly to the fast adoption of the service paradigm [20]. The standardization frenzy that ensued, while disorganized, has also galvanized the field where the technology is now widely used. The flagship technology and embodiment of service-oriented computing is web services technology [30]. The introduction of web services has been key to the wide and early adoption of service-oriented computing in industry and lately in government. In a nutshell, a *web service* is defined as a functionality that can be programmatically accessible via the web [25]. A fundamental objective of web services is to enable interoperability among different software and data applications running on a variety of platforms [30]. Web services are increasingly being adopted to access data and applications across the web [30]. This has been largely the result of the many standardization efforts to describe, advertise, discover, and invoke web services [22].

Web services provide an efficient vehicle for users to access the functionalities available on the web [30]. The development of web services has so far mostly been the result of standardization bodies usually operating on a consensus basis and driven by market considerations. In this context, innovation and long-term deployment issues are not usually of primary concern. Because of the global nature of the web, the standardization process has so far been very fragmented, leading to competing and potentially incompatible web service standards. Governments and commercial organization have meanwhile invested very heavily in web services technologies. These investments have resulted in a fast-growing number of web services being made available. The prevalent business model will, in all likelihood, include a autonomous and competing communities of web service providers vying for consumers attentions. It is, however, important that these investments produce the expected results and soon for the area to really make an impact. One-key impediment has been the lack of any rigorous and systematic methodology for managing the entire life cycle of web services that would include delivering, selecting, optimizing, and composing services. This needs to take place within a secure, trustworthy, and privacy protecting environment. We call the resulting system a Web Service Management System (WSMS). In summary, a WSMS is a comprehensive framework that provides an integrated view on the management of web services [30], including *automatic service composition*, *service query optimization*, *service privacy preservation*, *service trust*, and *change management*. Service composition is concerned with the automatic selection and integration of individual web services to provide value-added and personalized composite services [16]. Service query and optimization are concerned with the ability to model queries and provide an optimization framework suited for web services [29]. Service security and privacy preservation are concerned with ensuring that interactions with web services are conducted in a secure fashion, while sensitive information can be preserved as required [22]. Service trust is concerned

- *A. Bouguettaya is with the School of Computer Science and Information Technology, RMIT, GPO Box 2476, Melbourne 3001, Victoria, Australia. E-mail: athman.bouguettaya@csiro.au.*
- *Q. Yu is with the Department of Information Sciences and Technologies, Rochester Institute of Technology, 102 Lomb Memorial Drive, Rochester, NY 14623-5608. E-mail: qi.yu@rit.edu.*
- *X. Liu is with the Department of Computer Science, Rochester Institute of Technology, 102 Lomb Memorial Drive, Rochester, NY 14623-5608. E-mail: xl@cs.rit.edu.*
- *Z. Malik is with the Department of Computer Science, Wayne State University, Suite 200.3, Floor 14, 5057 Woodward Avenue, Detroit, MI 48202. E-mail: zaki@wayne.edu.*

Manuscript received 24 July 2009; revised 22 Dec. 2009; accepted 25 Feb. 2010; published online 6 Aug. 2010.
For information on obtaining reprints of this article, please send e-mail to: tsc@computer.org, and reference IEEECS Log Number TSC-2009-07-0183.
Digital Object Identifier no. 10.1109/TSC.2010.36.

with establishing trust among different and autonomous web services so that they can determine to which extent they can trust other services before interacting with them [12]. Change management is concerned with efficiently managing changes during the life-cycle of long-term composed web services [11].

In this paper, we use web services as a framework for implementing and deploying *Digital Government (DG)*. As the web has redefined the way users interact with businesses and government agencies, the DG research has received considerable attention nowadays [1]. Millions of people worldwide have adopted the web as their primary means for conducting many of their daily activities. According to a recent *Nielsen Online* report, the US "web population" has hit a milestone with nearly 75 percent or 226.3 million Americans accessing the Internet from home.[1] As part of an effort to improve government-citizen interactions, government agencies are now providing a wide spectrum of online services that implement the long-awaited DG. These e-government services usually span several application domains, including social programs, healthcare, voting, tax filing, etc. A 2004 study states that 97 million Americans have used government agency websites.[2] This study shows that people use government websites to access information, complete transactions, and are part of the process of government decision making [14].

Building a DG infrastructure entails a number of policy and technical challenges. The major issues that must be addressed for successful deployment of most DG applications include: data integration, interoperability of services, privacy and security, user interface, and user accessibility [3]. Government generally consists of large and complex networks of institutions and agencies, which often need to interoperate to create value for citizens. Government agencies data are typically distributed over a large number of autonomous and heterogeneous databases with an equally large number of platform-dependent applications [1]. Data and application integration thus require most attention to realize the DG vision. Moreover, complex applications involving coordination among various DG services and agencies need an integrated framework to efficiently access and manipulate the offered service functionalities. This requires a systematic support of query facilities. In addition, e-government applications would have to be built for average citizens that have no a priori special computer skills. Therefore, successful deployment of DG applications requires an efficient framework that accommodates autonomy, bridges heterogeneity, and integrates data and applications in the most useful and homogeneous way [3].

In this paper, we describe a WSMS called *WebSenior* which is a service-oriented digital government system that aims at providing services to senior citizens. Adopting web services in DG enables government agencies to: outsource from other DG services, compose existing services to provide value-added services, effectively handle privacy issues, and provide adaptive web-based user interface. WebSenior is a collaborative effort between Virginia Tech and the Virginia Department for the Aging (VDA). The VDA subcontracts with Area Agencies on Aging (AAAs) to offer services to communities throughout the state. AAAs work with public and private organizations to help seniors and their families find the services and information they need. One of the biggest barriers to providing services to customers is the lack of integrated information systems. The VDA and its partners offer many and diverse services. The challenge is how to get these varied systems to cooperate and share information. WebSenior wraps the legacy systems by using web service technologies. It provides an integrated service framework that achieves a seamless cooperation among government agencies to provide prompt and customized services to senior citizens. We summarize the key components of WebSenior, which constitute the major contributions of this paper as follows:

- *Service composition:* This component performs three tasks: *checking composability*, *checking soundness*, and *automatically composing services*. The first issue when defining a composite service is whether its component services are *composable*. We present a set of composability rules to ensure the feasibility of the composed services. Another important issue is whether a composite service provides an *added value*. We present the concept of composition templates, based on which the *soundness* of a composite service can be evaluated. Finally, we propose a matchmaking algorithm for automatic composition of web services.
- *Service Optimization:* This component performs a "user-centric" optimization that selects the composition plan with the best quality based on users' preferences. First, we define a score function to evaluate a composition plan. In the score function, a weighting mechanism is adopted to express users' preferences over different quality parameters. A set of aggregation functions is presented to compute the quality of a composition plan that consists of multiple operations. Second, we present two optimization approaches to find the best plan: *exhaustive* search and *greedy* search.
- *Service Privacy Preservation:* This component consists of two subcomponents that protect the sensitive information of users: *Privacy-Preserving Data Filter (DFilter)* and *Privacy Profile Manager (PPM)*. DFilter ensures that a user can access the information only if she has the requested credential. It first checks whether the service requester is authorized to access the requested information using the credential received with the query. It then rewrites the query to ensure that all the privacy constraints are enforced. PPM enforces privacy in a finer granularity than DFilter. It maintains a repository of privacy profiles that store individual privacy preferences.

The remainder of this paper is organized as follows:. In Section 2, we describe our digital government application. In Section 3, we present the proposed WSMS and identify the key components that are implemented in WebSenior. In Section 4, we present the implementation of WebSenior.

1. http://www.nielsen-online.com/press_fd.jsp?section=pr_netv&nav=3.
2. http://www.pewinternet.org/pdfs/PIP_E-Gov_Report_0504.pdf.

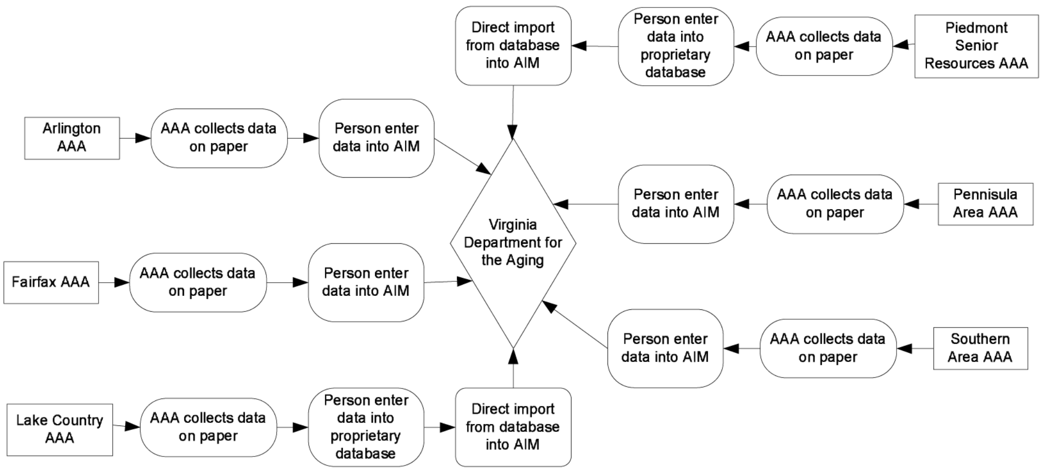

Direct import from database into AIM
Person enter data into proprietary database
AAA collects data on paper
Piedmont Senior Resources AAA
Arlington AAA
AAA collects data on paper
Person enter data into AIM
Virginia Department for the Aging
Person enter data into AIM
AAA collects data on paper
Pennisula Area AAA
Fairfax AAA
AAA collects data on paper
Person enter data into AIM
Person enter data into AIM
AAA collects data on paper
Southern Area AAA
Lake Country AAA
AAA collects data on paper
Person enter data into proprietary database
Direct import from database into AIM

Fig. 1. Reporting mechanisms between the AAAs and the VDA.

In Section 5, we describe several scenarios to showcase how WebSenior is used. In Section 6, we give an overview of some related work and identify some important future trends for web service technologies. We conclude in Section 7.

2 Application: Services for Senior Citizens

In this section, we describe the environment and interactions of our *Digital Government* application: WebSenior. The major objective of the project is to provide an interoperable infrastructure for the information systems of the VDA and AAA, which is able to provide prompt and customized service to senior citizens using the web.

The potential benefit of DG for senior citizens is of particular interest for two reasons. First, the population segment over 60 years old is growing dramatically in the US, and seniors' needs for assistance increase with age. It is estimated that 31.2 percent of seniors age 80-84 require assistance with everyday activities, including access to health-related, social, and welfare programs. The percentage increases to 49.5 percent for those over age 85 [13]. Second, seniors represent one of the fastest growing segments of web users. To capitalize on this trend, many federal, state, and local agencies now offer online forms to access e-government services dedicated to seniors' needs. Unfortunately, such efforts have so far had little effect in the lives of seniors in terms of their receiving better care and services. Accessing public services poses bureaucratic challenges that can be daunting in any case, but seniors are challenged in ways that are fundamentally different from other age groups. WebSenior is designed to automatically deliver e-government services that are customized for seniors.

The focus of WebSenior is on leveraging web services and related standards to provide: 1) a rich set of services where they can find specific existing resources and value-added services that are the result of service composition, 2) optimized service query infrastructure for combining accesses to diverse service providers, and 3) a systematic approach that helps protect users' privacy while exchanging sensitive information when accessing web services.

2.1 Overview of VDA and AAA

VDA oversees all state programs using funds provided by the federal Older Americans Act and the Virginia General Assembly. VDA's mission is to foster the dignity, independence, and security of older Virginians by promoting partnerships with families and communities. AAAs contract with VDA to provide services for elder Virginians and their families in communities throughout Virginia. The majority of the VDA's work come from managing these AAAs. This includes managing the contracts, auditing them, providing news and policy changes through a newsletter, and through a monthly database upload to the VDA.

VDA and the 25 AAAs deploy their databases in great variations. The 26 databases are from different vendors, running on various platforms and storing data in different formats. The VDA has adopted the Aging Information Management (AIM) System to track and report the aging service to the National Aging Program Information System (NAPIS) annual reporting requirements. AAAs collect data of services and store into their databases. At the end of every month, officers in AAAs refine the data, convert it into AIM compatible format, and send to VDA. Fig. 1 depicts the data flow between six AAAs and VDA. The potential data accuracy problem is illustrated in the figure. Due to the data lost during conversion, the audit in VDA found out many inaccuracies and missing or incomplete information. Furthermore, coping with the task of permanently monitoring the operation of the 25 AAAs is fairly expensive.

The AAAs have two functions: assessing seniors' cases to determine their needs and providing actual services to seniors. The services provided by the VDA to seniors through the AAAs include: adult day care, meal delivery, homemaker, legal assistance, transportation, etc. An AAA may use different funding sources to provide these services to senior citizens. For example, an AAA may get funds from the Department of Rehabilitation Services (DRSs) to provide some types of transportation services. The cooperation of AAA is thus not only limited to the VDA.

Providing an integrated view on web services benefits both VDA and AAAs. For example, VDA will get a more accurate audit report if it can interoperate with AAA's database system in real time. However, the autonomous, heterogeneous database systems of AAAs and VDA prevent these and similar interactions' efficiency. We use the web Services technology to provide a comprehensive and efficient solution which requires no change to the current database systems in AAAs. Using web services, the audit and monitoring process can be carried out dynamically. Meanwhile, AAAs still keep their independence and privacy.

2.2 Senior Citizens and Services

Senior citizens apply for various services through their AAA. The AAAs are responsible for assessing the needs of older Virginian within their respective planning and service areas (PSAs). Senior citizens can get eligible services, such as *meal delivery* and *transportation* services, after officers in AAA approve their assessments. However, the cumbersome paper work prevents the prompt interactions between senior citizens and service providers. Moreover, due to the complexity of the system, only experienced officers can access the requirements of senior citizens in an effective manner.

A 2004 report on older Americans and the Internet shows that the percentage of seniors accessing the Internet has jumped by 47 percent between 2000 and 2004.[3] They spend more time on surfing and looking for helpful information (such as healthy and medical information) from the web. However, senior citizens often have special mental and/or physical disabilities. Their specific needs, lack of knowledge, and limitations thus present challenges in accessing the desired services through the web. The prevailing web page design philosophy—one size fits all—does not support customization for the distinct requirements of senior citizens with different disabilities. In WebSenior, we cater to the specific needs of senior citizens by providing customized access to the services with their desired quality.

2.3 Senior Citizen Services Scenario

In this section, we describe a scenario that illustrates the challenges seniors face in obtaining social services from local government agencies.

Let's say that Mary, a handicapped and visually impaired retiree, wants to receive social services from her county's AAA. Typically, she would first have to travel to the AAA office for an interview. An agency caseworker would assess Mary's needs and try to find the best matches for them among a large number of potential services. Let's say the caseworker, John, determines that Mary might qualify for the following services: FasTran (transportation for the elderly and handicapped), Meals on Wheels (meal delivery), and Senior Activity Center (a facility that offers seniors a variety of activities). He advises Mary of the information she needs to submit to these organizations before she can receive their services. He might also help her prepare and transmit the information through various communication media: e-mail, snail mail, fax, and phone. In some cases, Mary might have to personally visit the agency offering the service. In all cases, the process is ultimately manual, and processing delays are more the rule than the exception. Furthermore, under the current system, if Mary decided to move to another county, the case worker at that county's AAA would have to initiate the same manual, error-prone process.

3. http://www.pewinternet.org.

As this scenario indicates, the current social services system suffers from intrinsic inefficiencies. This scenario becomes even more realistic when we consider that seniors are generally expected to identify the necessary government services they want and to be aware of the service eligibility requirements. For example, seniors can request transportation only if they are registered with an AAA, and their eligibility for a given transportation service, such as FasTran, depends specifically on where they live.

Our partnership with the VDA aims to give seniors easy and customized access to services through the web. For example, let's say that Mary wants to visit a senior activity center for lunch and she needs transportation to get there. Fig. 2 shows the online web-based process that Mary would follow. She would access WebSenior from a computer in her home, and the system would automatically return a composite service that best fits Mary's needs. For that purpose, it uses Mary's personal information stored in a user profile and government rules and regulations defining eligibility conditions for a service. WebSenior would transparently return results for the nearest senior activity center and a convenient transportation service in web pages appropriate to Mary's visual capacity (for example, in bigger fonts). Details on WebSenior follow.

3 WebSenior: A WSMS for Service Delivery to Senior Citizens

WebSenior is a WSMS for providing services to senior citizens in collaboration with VDA and AAAs. It integrates a set of key components: a service composition component, a service privacy component, and a service optimization component. These components enable seamless cooperation and coordination among three groups of users, VDA officers, AAA officers, and senior citizens, and provide prompt and customized services to senior citizens through web.

3.1 Dynamic Service Composition in WebSenior

Different senior citizens may make different service requests based on their particular situations. Therefore, it is important that WebSenior be adaptive to the personalized requests. The service composition component enables *customized* delivery of services, which is particularly suitable for e-government applications. Since it is impossible to predict all possible service requests, WebSenior needs to compose web services on demand and in a dynamic fashion.

3.1.1 Composability Model for Web Services

A major issue when defining a composite service is whether its component services are *composable* [16]. For example, it would be difficult to invoke an operation if there is no mapping between the parameters requested by this operation (e.g., data types and number of parameters) and those transmitted by the client service. In this section, we identify a set of *composability rules* to compare *syntactic* properties of web services. These rules include: 1) *mode composability* that compares operation modes; 2) *binding*

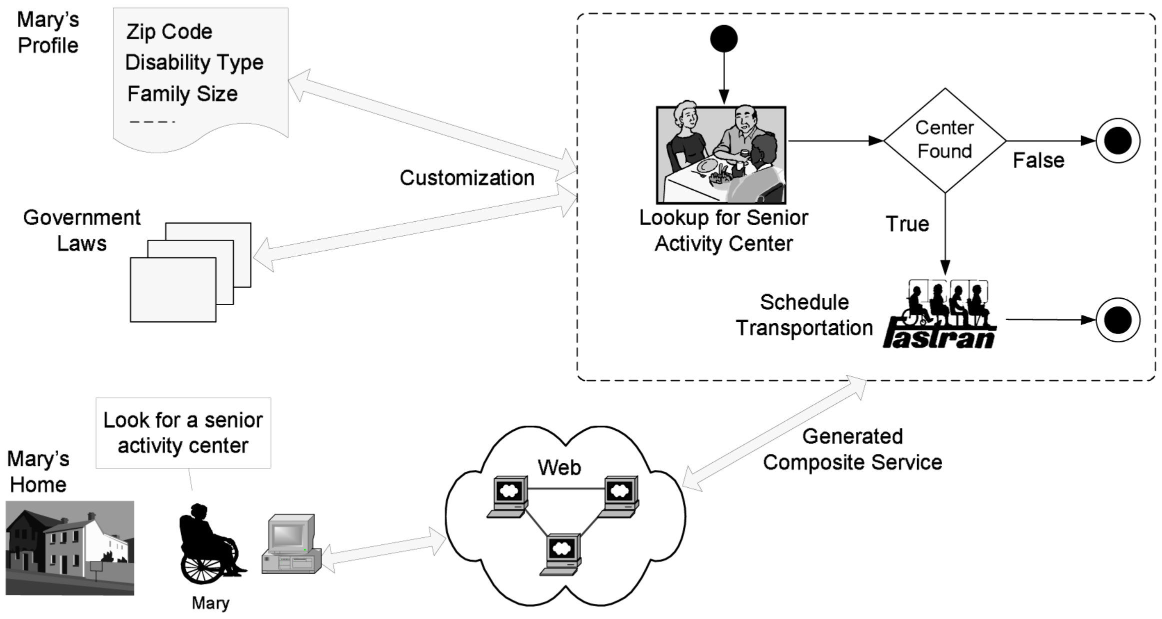


Fig. 2. WebSenior process for delivering e-government services.

composability that compares the binding protocols of interacting services; and 3) *message composability* that compares the number of message parameters, their data types, business roles, and units.

In our model, clients and servers refer to *outsourcing* and *outsourced* services, respectively. We adopt the approach from XLANG [17], WSFL [10], and BPEL4WS [2] standards where web services at both sides (client/server) are defined in WSDL augmented with semantic capabilities. Similarly, we assume that each operation on the server side has at most one matching operation at the client side and vice versa [2], [10], [17].

Mode and binding composability. For web service interactions to take place, operations at client and server sides must have "dual" modes [10], [17]. A *notification* operation at one service must be connected to a *one-way* operation at a partner service. Similarly, a *solicit-response* operation maps to a *request-response* operation at a partner service.

Assume now that two web services are communicating through operations that are mode compatible. Since these web services may support different binding protocols (e.g., SOAP, HTTP, or MIME), it is important to insure that they "understand" each other at the message format and protocol level. At least one of the protocols expected by a web service must be supported by the other. For example, it would be difficult for a service that expects to receive messages in MIME protocol to interact with another service that formats its messages in HTTP. The following rule, called *binding composability*, checks that web services support at least one common binding protocol.

Message composability. Interactions between web services involve the exchange of messages. A message consists of one or more parameters, each parameter having a certain data type. Hence, it is important to check that the data types of the parameters sent by a service are compatible with the data types of the parameters received by its partner. We consider two primary data type compatibility methods: *direct* and *indirect compatibility*. Two parameters are *directly compatible* if they have the same data type. A parameter p is *indirectly compatible* with a p' if the type of p is derived from the type of p'. For example, a parameter with a `positiveInteger` or `short` type is indirectly compatible with an `integer` parameter. Note that contrary to direct compatibility, indirect compatibility is asymmetric.

We extend the notion of data type compatibility to messages as follows: A message M is *data type compatible* with a message M' if every parameter of M is directly or indirectly compatible with a parameter of M'. Note that not all parameters of M' need to be mapped to the parameters of M. The rationale is that an input message of a service operation may use only a subset of the parameters sent through an output message of a "dual" operation.

Assume now that a parameter p is compatible (directly or indirectly) with a parameter p'. For p and p' to be mapped together, they must also have compatible semantics. For example, a total price should not be mapped with a price before taxes. Similarly, a price in US dollars should not be mapped to a price in Yen. To this end, we compare the units and business roles of p and p'. Both parameters should have the same units and business roles. The *message composability* rule compares the input and output messages of every pair of operations. The idea is to check that each input of an operation is data type compatible with the output of the other operation. The input's unit and business role should be the same as the output's unit and business role, respectively. This means that the parameters of each input message map to all or some of the parameters contained in the output message of the other operation.

3.1.2 Composition Soundness

Another important aspect to consider in service composition is whether combining a set of services in a specific way provides an added value. For example, it would probably be

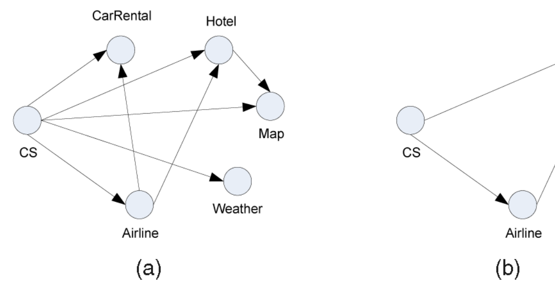


Fig. 3. Examples of composition templates.

"unusual" to combine a *lemon check* service with a *hotel* service. However, combining *airline* and *hotel* services would provide a *travel preparation* composite service. The idea is to define a rule, called *composition soundness*, to test whether composite services are *sound*. By *sound*, we mean that the way component services are composed provides an added value. To this end, we introduce the notion of *composition templates*. These are graphs built using *precedence relationships*.

A *composition template* is associated with each composite service and gives the general structure of that service. It is formally defined as follows:

Definition 3.1 (Composition template). *A composition template is a directed graph (V, E), where V is a set of service category names and E is a set of edges. A special vertex corresponds to the composite service and has the special value "CS." An edge* $(v_i, v_j) \in$ *E means that a service of category name* v_i *precedes a service of category name* v_j.

An example of a composition template is given in Fig. 3a. Composition templates are used to compare the values added by different compositions. For example, consider the template depicted in Fig. 3b. This template is a subgraph of the template depicted in Fig. 3a. This means that the second composite service would provide a subset of the functionalities offered by the first one. For example, it does not provide "carRental," "map," and "whether" operations.

To check whether a composition of services is *sound*, we define the notion of *stored templates*. *Stored templates* are divided into two groups. The first group includes templates that are predefined by domain experts (e.g., cars, travel, and computers). For example, the travel industry would agree that a *travel preparation* composite service combines *airline*, *hotel*, and *car rental* services. The second group includes templates that are "learned" by the system. Each time a composite service is defined, the system stores its template in the repository. For example, assume that the composer defines a service whose template is depicted in Fig. 3a. If the template does not already exist in the repository, the system would store it for future use. Since stored templates inherently provide added values, they are used to test the soundness of composition plans.

Definition 3.2 (Composition soundness). *A composition of services is* sound *if its template is a subgraph of a* stored template.

Stored templates are different from *process templates* and *reference processes* used in [4] and [23], respectively. Indeed, process templates and reference processes are used as a priori "canvas" when defining composite services. In contrast, stored templates are used a posteriori to check the soundness of composite services, i.e., once they have been generated. It is important to note that the composition soundness rule is not used to determine if web services are composable. It is rather used to determine if composing a given set of services provides an added value. Even if a composition is not sound, composers have the flexibility to decide whether they are willing to consider such composition as "acceptable."

3.1.3 Automatic Composition of Web Services

Based on our composability model, we propose a *matchmaking* algorithm for the automatic composition of web services (see Table 1). The algorithm uses the aforementioned composability rules to generate *composition plans* that conform to composers' specifications. By *composition plan*, we refer to the list of component services and their interactions with each other (plugging operations, mapping messages, etc.) to form the composite service. Since the number of generated plans may be large, composers have the possibility to control the number of generated plans through the *nb_requested_plans* input.

For every pair of operations (op_{ik}, op_{jl}), the algorithm checks mode composability (line 10) and message composability (line 11). The *message_compatible()* function returns *true* or *false* depending on whether a message M_i is message compatible with M_j. The *sound()* function checks the soundness of the generated plan. Once a template has been computed for the generated plan, it is compared with stored templates. Note that as stated previously, composition plans are returned to composers even if they are not sound. Each iteration of the `while` statement generates a composition plan. The algorithm stops until the user-specified number of plans are generated.

3.2 Service Optimization in WebSenior

The purpose of the service optimization component is to select the best service(s) or their compositions to fulfill senior citizens' service requirements. Different service providers (e.g., senior centers and transportation companies) may compete to offer similar services. At the end of the matchmaking algorithm described in last section, multiple composition plans may be generated. These composition plans offer the same functionality, but they may be different in terms of the nonfunctional aspects. The concept of Quality of Web Service (QoWS) is emerging as a key feature in distinguishing between competing services. QoWS reflects the runtime and business requirements of web services, such as response time, availability, reliability, cost, and reputation. For example, senior citizens may take interest in the response time and the service reputation when selecting a meal delivery service.

The service optimization component performs a "user-centered" optimization. It is to find the composition plan with the best quality based on user preference. Since a composition plan typically consists of multiple operations, we first present a set of aggregation functions to compute the QoWS for a composition plan. They combine the QoWS parameters from multiple service operations. Table 2 gives

TABLE 1
Matchmaking Algorithm

```
(01) Input: $WS_i$, repository, nb_requested_plans {
(02)   nb_generated_plans = 0
(03)   matched = $\emptyset$
(04)   while nb_generated_plans $\leq$ nb_requested_plans do
(05)     { plan = $\emptyset$
(06)     for each operation $op_{ik} \in WS_i$ do
(07)       { found = false
(08)       for each service $WS_j$ from repository | category_compatible($C_{ik}$,$C_j$)
(09)                       and purpose_compatible($P_{ik}$,$P_j$) and ($B_i \cap B_j \neq \emptyset$) do
(10)         { for each operation $op_{jl} \in WS_j$ | ($mode_{ik}$ and $mode_{jl}$ are dual) and ($op_{jl} \notin$ matched)
(11)           if message_composable($in_{ik}$,$out_{jl}$) and message_composable($in_{jl}$,$out_{ik}$)
(12)           then { found = true
(13)                  plan = plan $\cup$ {($op_{ik}$,$op_{jl}$)}
(14)                  matched = matched $\cup$ {$op_{jl}$}
(15)                  break }
(16)           if found then break
(17)         } /* for in line (08) */
(18)         if $\neg$found
(19)         then { output("no matchmaking for",$op_{ik}$)
(20)                break }
(21)       } /* for in line (06) */
(22)     if $\neg$found then break
(23)     else if sound(plan)
(24)     else output(plan,"not sound" )
(25)     nb_generated_plans = nb_generated_plans + 1
(26)   } /* while in line (04) */ }
```

the details of these aggregate functions. An score function is then presented to evaluate the entire composition plan. Finally, we present two optimization approaches to find the best plan.

3.2.1 Score Function

Users' preferences over different QoWS parameters are specified by the relative importance of these parameters. We assign *weights*, ranging from 0 to 1, to each *QoWS* parameter to reflect the level of importance. This enables us to define the following score function F to evaluate the quality of the entire plan:

$$F = \sum_{Q_i \in Neg} W_i \frac{Q_i^{max} - Q_i}{Q_i^{max} - Q_i^{min}} + \sum_{Q_i \in Pos} W_i \frac{Q_i - Q_i^{min}}{Q_i^{max} - Q_i^{min}}, \quad (1)$$

where Neg and Pos are the sets of negative and positive *QoWS*, respectively. In negative (respectively, positive) parameters, the higher (respectively, lower) the value, the worse is the quality. W_i are the weights assigned by users to each parameter. Q_i is the value of the ith *QoWS* of the service execution plan obtained through the aggregate functions from Table 2. Q_i^{max} is the maximum value for the ith *QoWS* parameter for all potential service execution plans and Q_i^{min} is the minimum. These two values can be computed by considering the operations from service instances with the highest and lowest values for the ith *QoWS*.

3.2.2 Generating the Best Composition Plan

By using the score function, service optimization is to find the plan that maximizes the value of F. We present two approaches for finding the best plan: *exhaustive* search and *greedy* search [29].

The exhaustive search enumerates the entire space of composition plans. Suppose that a service request needs to access M web services (e.g., transportation, congregate meals, etc.). The composition plans are generated by composing M corresponding services. We assume that there are N service providers that compete to offer each service. Therefore, the complexity of the exhaustive search is $O(N^M)$, which is exponential.

The greedy search achieves the polynomial complexity by using a *divide-and-conquer* strategy. It generates an *optimal subplan* from each service through local search. It then combines these subplans to form the final plan. In order to apply the divide-and-conquer strategy, we take logarithms on the aggregation functions for reliability and availability. Specifically,

$$Reliability = \sum_{i=1}^{n} log(rel(op_i)), \quad (2)$$

$$Availability = \sum_{i=1}^{n} log(avail(op_i)). \quad (3)$$

TABLE 2
QoWS for a Composition Plan

QoWS parameter	Aggregation function
lat(plan)	$\sum_{i=1}^{n} lat(op_i)$
rel(plan)	$\prod_{i=1}^{n} rel(op_i)$
avail(plan)	$\prod_{i=1}^{n} avail(op_i)$
fee(plan)	$\sum_{i=1}^{n} fee(op_i)$
rep(plan)	$\frac{1}{n}\sum_{i=1}^{n} rep(op_i)$

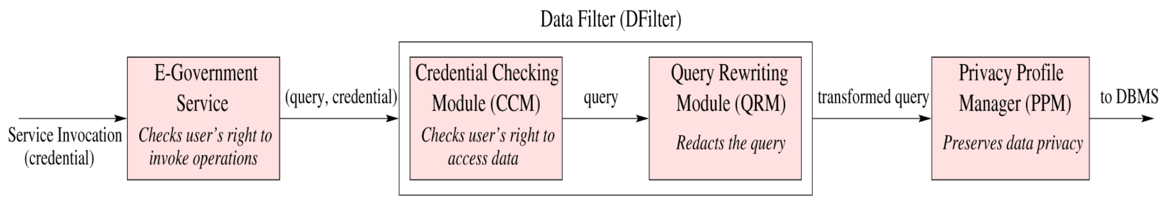

Data Filter (DFilter)
Service Invocation (credential)
E-Government Service
Checks user's right to invoke operations
(query, credential)
Credential Checking Module (CCM)
Checks user's right to access data
query
Query Rewriting Module (QRM)
Redacts the query
transformed query
Privacy Profile Manager (PPM)
Preserves data privacy
to DBMS

Fig. 4. The data filtering mechanism for privacy preservation.

These enable to express the score of the final plan as a linear combination of the scores from the subplans. Thus, the greedy search has a complexity of $O(N \times M)$.

3.3 Preserving Privacy in WebSenior

Privacy is a major issue that needs to be addressed in e-government applications. The sharing of information between different government agencies is dictated by the fact that an e-government middleware, while providing integrated services to citizens, must eliminate redundancy and reduce management effort and cost. These agencies may have different (and possibly conflicting) privacy policies. Moreover, citizens typically have different preferences with respect to their privacy. For example, senior citizens typically do not want to release their income or health-related information. The challenge is to achieve seamless interoperability between different government agencies without violating citizens' privacy.

In WebSenior, users submit their service requests along with their Digital Privacy Credentials (DPCs). When a service receives a request from a given user (see Fig. 4), it first checks that user has the necessary credentials to access the requested operation according to its privacy policy. For example, the service HealthForSeniors may require that updating the privacy profile of a citizen can be done only by that same citizen. If the request can be answered, the service translates the user's service request into an equivalent data query that is submitted to the appropriate government database.

When the query is received by the database, it is first processed by a privacy-preserving data filter (DFilter). The DFilter is composed of two modules: the Credential Checking Module (CCM) and the Query Rewriting Module (QRM). The CCM uses the credential received with the query to determine whether the service requester is authorized to access the requested information. If the credential authorizes access to only part of the requested information, the QRM redacts the query so that all the privacy constraints are enforced. For example, assume that a user issues a service request that is translated into the SQL query "select name, age, and income from MealsOnWheels. enrollees." If the request credential does not authorize access to enrollees' income, the QRM will delete the income field from the query before it is submitted to the database.

The PPM is responsible for enforcing privacy at a finer granularity than that enforced by the CCM. For example, the local CCM may decide that a given organization can have access to local information regarding a group of citizens' health records. However, a subset of that group of citizens may explicitly request that parts of their records should not be made available to third-party entities. In this case, the local PPM will discard those parts from the generated result. The PPM is a translation of the consent-based privacy model in that it implements the privacy preferences of individual citizens. It maintains a repository of privacy profiles that store individual privacy preferences. Requests made by citizens to update their privacy profiles are also handled by the PPM.

4 IMPLEMENTATION

Based on the architecture we described above, we organize *WebSenior* into four tiers: *user interaction management*, *service management*, *service ontologies*, and *basic web services*. The key components of the user interaction management tier are implemented using JSP/Servelet and deployed on a Tomcat web server. The service management tier is implemented using Java. The web services are developed on Systinet WASP Server, which is a complete platform for development, deployment, and management of web-service-based applications. It is worth to note that the ontologies are also implemented as web services and can be accessed by using the same way as the basic services. The architecture is illustrated in Fig. 5. Note that each AAA has its own database storing the information of services and all registered users, including senior citizens and officers. Also, the service transactions are recorded in the databases. VDA also has its own database, which contains basic user information mainly used for authentication. These databases are distributed on a set of Oracle 8i servers. Note that the system can easily be extended by adding other databases. New services can also be included with minimum overhead by registering themselves with the UDDI service registry.

4.1 User Interaction Management Tier

The functionality of the user interaction tier is to authenticate users, present service information, and record and analyze user's behavior. After identifying the user, the user interface will change adaptively according to user's static profile and his or her current operation behaviors. There are six components in this level: Adaptive User Interface Generator, User Authentication, User Behavior Monitor, User Behavior Analyzer, Request Handler, and User Profile Management.

- *Adaptive User Interface Generator*—The functionality of this component is to dynamically generate friendly user interface. The content of the user interface comes from the response of the request handler. The view of the web page is based on schemes (which is designed for different groups of seniors). Adaptive User Interface Generator gathers

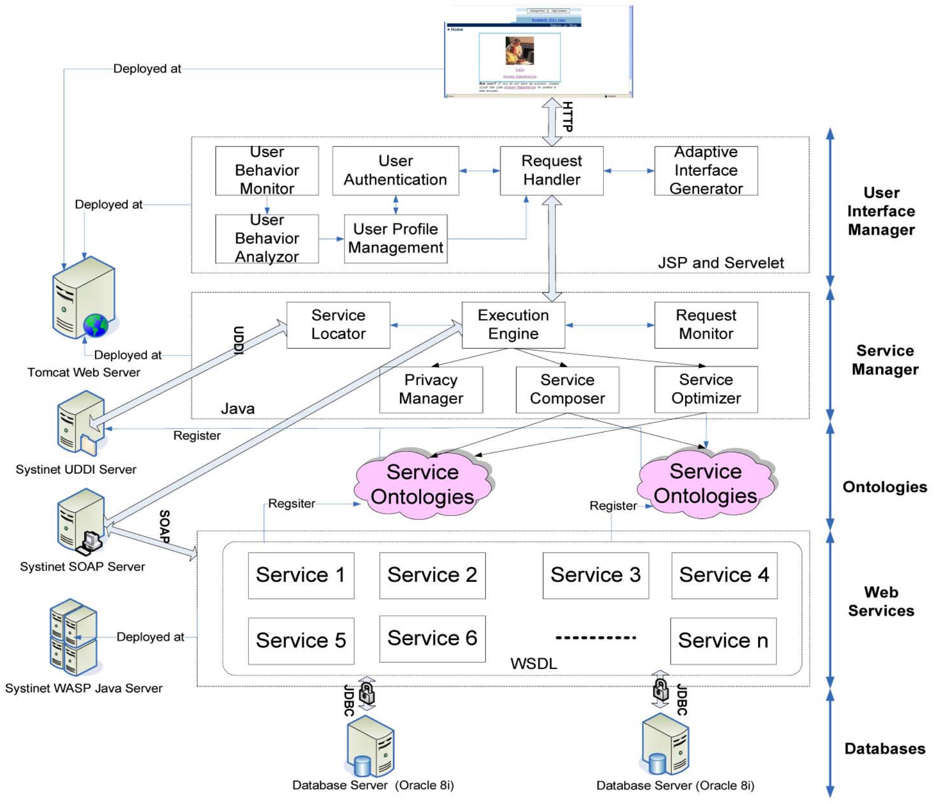


Fig. 5. The architecture of *WebSenior*.

the content from request handler, identifies the user's profile information, and then, selects the specific scheme. Based on the user's dynamic profile information, it organizes the content of the page and adjusts the display style for the user.

- *User Authentication*—User authentication component gets the login information from Request Handler and forwards that information to the corresponding AAA's local user information system. After getting the approval information of that user from AAA, User Authentication component assigns an identity to the user and requests User Profile Management component to allocate space to store user's dynamic profile information. After that, it forwards the identity to the Request Handler component. Otherwise, it just sends the deny message to the Request Handler.
- *Request Handler*—It is a business logic component. Seniors may not type efficiently and correctly. As a result, they may not input their query messages completely and accurately. Request Handler component gathers the input data from user, completes (corrects) the queries from users, combines more helpful information from User Profile Management, and sends them to the query optimizer. The integrated query may help the optimizer to perform efficiently. After it gets the response from optimizer, it processes the information and does transformation and filtering. It then combines users profile information and delivers the data to the Adaptive User Interface Generator.
- *User Behavior Monitor*—This component residents on the client side. Its main functionality is to track the user's operations on the computer. The component records the data, and aggregates and delivers it to the User Behavior Analyzer.
- *User Behavior Analyzer*—After gathering the data from the User Behavior Monitor, User Behavior Analyzer analyzes the data and generates conclusions based on the behavior evaluation metrics.
- *User Profile Management*—This component retrieves user's information from the AAA's user information database and stores it as user's static profile information. It also manages the dynamic profile information which is generated from User Behavior Analyzer component.

4.2 Service Management Tier

There are five major components for service management in the WebSenior system architecture, including service locator, service composer, service optimizer, execution engine, and execution engine. In what follows, we give a detailed description of each component:

- *Service locator*—Service locator retrieves the service descriptions from UDDI and sends them to other components. All the other three components may need the service descriptions to achieve their tasks. Therefore, they all need to contact the service locator to finish their tasks.
- *Service composer*—Service composer composes several web services to fulfill users' requests with value-added services. It needs to contact the service locator

to get the required service descriptions. It also needs to contact the ontology manager to get the ontology information.

- *Service optimizer*—Service optimizer aims to provide the best quality of service through execution of a set of web services. It first selects web services (single services and composite services) to generate an execution plan. The selection is based on a quality of service model to optimize the execution plan. The service optimizer then executes the plan and sends the result back to the user.
- *Execution Engine*—Execution engine invokes the services via SOAP.
- *Privacy Manager*—Privacy manager implements the privacy-preserving mechanisms described in Section 3.3.
- *Request Monitor*—Request Monitor can detect the unfulfilled requests and store them in the AAA's databases. AAA officers can keep track of these unfulfilled requests. Once resources are available, senior citizens can get the services immediately.

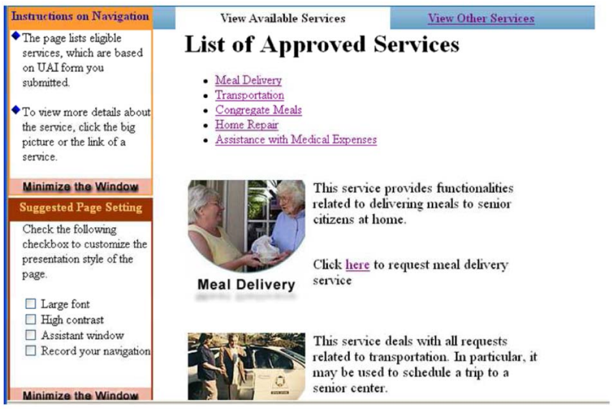
4.3 Service Ontologies Tier

We use ontologies to organize services based on their domain of interests. The ontologies are also implemented as web services and can be accessed by using the same way as the basic services. An ontology is a common understanding in a particular domain of interest. It defines a set of terms shared by all parties in this domain. Services need to register with one or more ontologies. This would greatly facilitate the service discovery process, which can only search ontologies containing services of interest. Based on the services we implemented, we created 25 ontologies. Due to space limitations, we describe only five of them, including meals, transportation, home repair, medical assistance, and management.

- *Meals ontology*—This ontology covers two major categories of meal services: meal delivery and congregate meals. Both of these two categories consist of a large number of service providers, which offer different types of foods and have different time schedules, etc.
- *Transportation ontology*—This ontology contains the transportation services offered by all the AAAs. This may include "Fastran," "Vans Shuttle," and "Taxi services."
- *Home repair ontology*—This ontology contains services that deal with home repair issues.
- *Medical assistance ontology*—This ontology contains services that assist senior citizens in coping with their medical costs.
- *Management ontology*—This ontology contains services that help AAA officers, and VDA officers manage users and services. This includes registration assessment, request assessment, and auditing.

4.4 Basic Web Services Tier

We have implemented 25 basic web services for senior citizens. Due to space limitations, we describe only five services for illustration purposes, including MealsDelivery, CongregateMeal, Transportation, ResidentalRepairAndRevonation, and Medical Assistance.



Instructions on Navigation

- The page lists eligible services, which are based on UAI form you submitted.
- To view more details about the service, click the big picture or the link of a service.

Minimize the Window

Suggested Page Setting

Check the following checkbox to customize the presentation style of the page.

- ☐ Large font
- ☐ High contrast
- ☐ Assistant window
- ☐ Record your navigation

Minimize the Window

View Available Services | View Other Services

List of Approved Services

- Meal Delivery
- Transportation
- Congregate Meals
- Home Repair
- Assistance with Medical Expenses

Meal Delivery

This service provides functionalities related to delivering meals to senior citizens at home.

Click here to request meal delivery service

This service deals with all requests related to transportation. In particular, it may be used to schedule a trip to a senior center.

Fig. 6. View eligible services.

- *MealsDelivery*—This service provides functionalities related to delivering meals to senior citizens at home.
- *CongregateMeal*—This service provides functionalities related to providing meals to citizens in congregation.
- *Transportation*—This service deals with all requests related to transportation. In particular, it may be used to schedule a trip to a senior center, a shopping center, or a doctor's office.
- *ResidentialRepairAndRenovation*—This service may be invoked to help senior citizens with home repair issues. It may be invoked *before* or *after* the repair. In the first case, the AAA assists in finding appropriate repair agents to help the citizen. In the second case, the AAA helps the citizen in paying for repairs that he or she has himself/herself previously paid for.
- *Medical Assistance*—This service is used to assist senior citizens in coping with the different types of medical cost.

5 USING WEBSENIOR

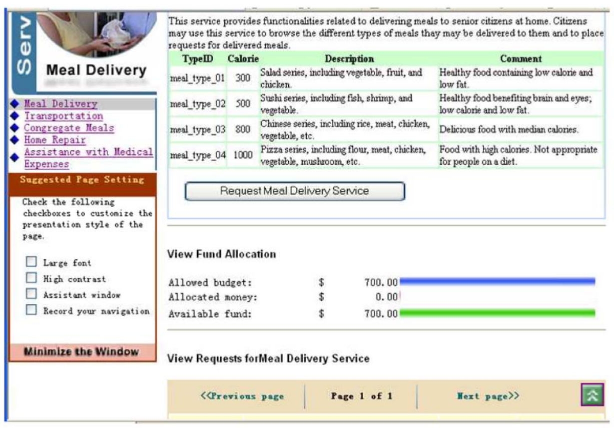
In what follows, we describe several scenarios to showcase the use of WebSenior. We use the experience of a senior citizen, Charlie, as our running examples in the scenarios. There are 25 services that have been implemented in WebSenior. We assume that Charlie is already registered with the system, and has submitted the assessment forms that made him eligible to request and receive services. For brevity, we only describe the most representative scenarios.

5.1 Requesting Basic Web Services in WebSenior

In this section, we use meal delivery service as an example to give a detailed introduction of how to request a basic service via WebSenior.

First, Charlie needs to log in in the system by using his username and password obtained during the registration process. Then, he will be directed to the "View Available Service" page that contains all the services that Charlie is eligible for, such as Meal Delivery and Transportation.

On the "View Available Services" page, Charlie can begin to request his desired services (see Fig. 6). For example, if he clicks on the Meal Delivery service, he will be directed to the meal delivery service page (see Fig. 7).



This service provides functionalities related to delivering meals to senior citizens at home. Citizens may use this service to browse the different types of meals thay may be delivered to them and to place requests for delivered meals.

TypeID	Calorie	Description	Comment
meal_type_01	300	Salad series, including vegetable, fruit, and chicken.	Healthy food containing low calorie and low fat.
meal_type_02	500	Sushi series, including fish, shrimp, and vegetable.	Healthy food benefiting brain and eyes; low calorie and low fat.
meal_type_03	800	Chinese series, including rice, meat, chicken, vegetable, etc.	Delicious food with median calories.
meal_type_04	1000	Pizza series, including flour, meat, chicken, vegetable, mushroom, etc.	Food with high calories. Not appropriate for people on a diet.

Check the following checkboxes to customize the presentation style of the page.

View Fund Allocation

```
Allowed budget:      $   700.00
Allocated money:     $     0.00
Available fund:      $   700.00
```

View Requests forMeal Delivery Service

Fig. 7. Meal delivery service.

Charlie can view the detailed description of the delivered meals, including types, calories, descriptions, and comments. Below the description of this service, Charlie can also view the fund information and his previous submitted service requests. He can check the submitted time and the current status of these requests.

After reading this, Charlie can make a request by clicking the "Request Meal Delivery" button. Charlie will then be directed to the "Request a Meal Delivery Service" page. On this page, Charlie can view a detailed description of this service. To make the request, Charlie needs to fill some information, including types of meals, delivered time, start date, end date, frequency, and address. The meal will be delivered to Charlie's home by default. Charlie can also specify other address if he wants the meal to be delivered to another address other than his home. After checking everything is correctly input, Charlie can submit the request.

The confirmation result contains the request ID, the type of the meal, time to deliver, address to deliver, and the frequency for delivery. Charlie can continue to make a new service request by clicking on the "make a new request" button. He can also check his requests by clicking of the "view your submitted requests." Charlie can see the detailed information of all the requests he has submitted, including the request ID, submission time, and request status. The request status can be waiting to be processed, approved, denied, and partially approved. Since Charlie just submitted his request, the status of the request is waiting to be processed, which means that this request needs to be processed by the AAA officers. The AAA officers can also use WebSenior to process all the requests from senior citizens.

5.2 Service Composition and Optimization in WebSenior

Assume that Charlie also needs the congregate meal service in addition to the meal delivery service. Since he has Parkinson's disease, Charlie cannot drive to the senior service center. Therefore, he also needs a transportation service that works together with congregate meal service. To request such a service package, Charlie needs to follow several similar steps as to request the meal delivery service.

After login, Charlie needs to choose Congregate Meals Service. By clicking on the token that represents this service, Charlie will be directed to the Congregate Meals Service page. Charlie can read the description of this service and make a request by clicking on the "Request Congregate Meals" button.

Charlie will then come to the Request Congregate Meal Service page. He also needs to specify the time and frequency he prefers to have the meal. Finally, Charlie has a choice of requesting the congregate service with transportation or without transportation. Due to his Parkinson's disease, Charlie chooses the transportation option and gives the pickup address. The service provides a round trip from Charlie's home to the senior service center by default. Charlie can also specify a starting address other than his home. He then clicks on the "Submit Request" button. The two services are automatically composed and the result of the composition plan is a complete schedule for the visit. More specifically, there will be three output parameters from the congregate meal services: meal start time, meal end time, and location of the senior service center. These will serve as the input parameters for the transportation service. In particular, the meal start time and center location will be the drop off time and address for the departing trip, and the meal end time and center location will be pickup time and address for the returning trip.

Since there may be more than one senior centers available to provide the congregate meal, the best provider can be selected based on the distance to Charlie's home, the quality of the food, and the service reputation of the center based on Charlie's preference.

6 Related Work and Future Trends

Standards have so far been the key enablers of web services. The intensive standardization support reflects the strong industry backing of web services as a key mechanism for deploying programmable functionalities on the web [30]. However, the field of web service research is fragmented and still in its infancy. In this section, we discuss some work that is most relevant to the WebSenior project and the representative solutions. These include semantic web service technologies, service optimization, service composition, and some existing e-government web service projects. We also put forward several research trends that would be of significant importance for the future evolution of web service technologies. These include change management of composite services, web service mining, and M-Services.

DARPA Agent Markup Language-Service (DAML-S) is a major effort aimed at enabling the semantic description of web services (www.daml.org/services). However, DAML-S gives little support for the business process semantics of web services. For example, it does not allow the specification of pre- and postoperations for automatically generating composite services. Nor does it explicitly define notions of behavior and business logic. The Web Service Modeling Ontology describes various aspects related to Semantic web services (www.wsmo.org). WSMO uses the Web Service Modeling Framework as a starting point and refines it by developing a formal ontology and language. Like DAML-S, WSMO provides little or no support for specifying interoperational relationships.

In [24], a WSMS is proposed to enable optimized querying over web services. It incorporates web services into the traditional Select-Project-Join queries and treats them as a type of expensive predicates [9]. An algorithm is proposed to arrange service calls into a pipelined execution plan. The optimization is "performance centered" that focuses on reducing the total running time. In [19], a query model is proposed that offers query optimization functionalities for web services. The query model consists of three levels: *query level, virtual level*, and *concrete level*. The query model uses the predefined mapping rules to map relations defined at the query level to virtual operations defined at the virtual level. Users can thus directly use relations to query web services. In [31], a composite service optimization approach is proposed based on several quality of service parameters. Composite services are represented as a state chart. The optimization problem is tackled by finding the best web services to execute a composite service in the form of a linear programming problem.

A recent trend in automatic service composition is to use AI planning techniques. The composition engine treats service composition as a planning problem. Ideally, given a user's objective and a set of web services, a planner would find a collection of service requests that achieves the objective. For example, Simple Hierarchical Ordered Planner 2 (SHOP2) adopts the Hierarchical Task Network concept as a planning methodology [6]. The planning system decomposes tasks into smaller and smaller subtasks, until it finds primitive tasks that can be performed directly. AI planning techniques differ from our approach in two main aspects. First, AI techniques are knowledge based, while our approach is policy- and rule-driven. Second, AI techniques provide little support for web service standards, while our approach is built on standards such as SOAP, UDDI, and WSDL. Ninja (a framework for network services) introduces a technique called automatic path creation to support service composition [7]. When APC receives requests for composite service execution, it creates a path that includes a sequence of operators. The operators perform computation on data and the connectors that provide data transport between operators. Ninja focuses mostly on fault tolerance by replicating services on multiple workstations. Its operator functional classifications for automating selection are limited. It doesn't consider the business logic and relationship between operations. SWORD (a developer toolkit for web service composition) is a technique that uses a rule-based expert system to automatically determine whether a desired composite service can be constructed using existing services. SWORD doesn't focus on relationships between web services or customized composite web service generation based on user profiles. Other techniques for composing web services include WISE, eFlow, and CMI [15]. These techniques mostly provide constructs for specifying composite services. They provide little support for automatic web service composition.

To our knowledge, WebSenior is among the first efforts to provide support for e-government web services. The Organization for the Advancement of Structured Information Standards (OASISs) has an e-government technical committee. The TC is a forum for governments to express their requirements with respect to XML-based standards. It provides a mechanism for creating best practice documents and promotes the adoption of OASIS specifications. The US federal government published a draft document based on a TC recommendation to enhance interoperability by adopting a common data search standard. However, TC efforts to support e-government web services are still in their infancy. MyNJbusiness, another project that applies web service technologies for e-government [5], deals with customized service delivery in the context of the US Small Business Development Center. It specifies governmental regulations as a condition-action rule base to identify component services. The rule base is a collection of statements about regulatory, functional, geographical, and temporal regulations. However, MyNJbusiness doesn't deal with the business logic of operations and pre- and postoperation relationships.

In what follows, we identify some important future trends for web service technologies. These include change management of composite services, web service mining, and M-Services.

6.1 Change Management of Composite Services

Fully realizing composite services lie in providing support to improve their adaptability to the dynamic environment, i.e., to deal with changes during the lifetime of a composite service. Due to the dynamic nature of web service infrastructure, changes should be considered as the "rule" and managed in a structured and systematic way. Changes are usually introduced by the occurrence of new market interests, business regulation, new technologies, or some unexpected situations. They are classified into two categories: *top-down* changes and *bottom-up* changes. Bottom-up changes refer to those that are initiated by the outsourced web service providers. For example, a meal delivery service provider may change the functionality by adding a new operation for checking the delivery status, or a transportation service may suddenly be unavailable due to a unexpected network failure. Top-down changes refer to those that are initiated by the owner of the composite service. For example, the owner of the WebSenior may want to add a new functionality to enforce a new government policy. There are some existing standards providing related functionalities that can be adapted to dealing with changes of services. For example, Web Services Eventing (WS-eventing) and Web Services Notification (WSN) [18], [27] focus on providing and event-based framework that monitors the activities of web services. Once there is an event occurs in a service, it can be sent to the other services via message passing. By affiliating a change with an event, the change on one or more member services of a composite service can be propagated to the other component services. Nevertheless, the design focus of this framework is not for change management purpose. Furthermore, it does not support for dealing with top-down changes at all.

6.2 Web Service Mining

Web service mining aims to detect interesting and useful patterns from web services. It provides a more intelligent and transparent way to deliver high-quality web services.

Web service mining offers two advantages over service composition. First, since it is not restricted by composition conditions, service mining has more flexibility than service composition. Second, the discovery process is capable of finding unexpected service combinations. These combinations are not achievable through traditional composition techniques because the combined web services may not semantically compatible with each other in terms of functionalities. In addition, the combinations would be very useful because they are summarized from various previous experiences through a training process. Traditional data mining technology is an automated process of extracting structured knowledge from large volume of data [8]. It has achieved great success in various application areas, including medical diagnosis, financial analysis, and stock price prediction. However, extending data mining techniques toward web service mining needs to deal with several important issues. Web services are more complex objects than data. A service mining technology needs to equip with an effective service processing mechanism. In contrast to data, web services are dynamic and may change over time. Therefore, adaptability to dynamics would greatly affect the service mining performance.

6.3 M-Services

The significant advances in wireless technologies open a promising application area for web services. As a key extension of web services, *mobile services* (*m-services*) cater for the increasing population of mobile users. Specifically, *m-services* are a special set of web services that can be accessible by mobile hosts over wireless networks [21], [28]. They work together with mobile devices to offer *anytime/anywhere* accessible services. In contrast to web services with wired infrastructures, m-services are more suitable for time and location critical tasks [26]. For instance, a stock quote service can help users make quick response by providing timely quote prices. However, users may prefer desktops to cell phones or personal digital assistants (PDAs) when carefully preparing a travel package for vacations. The mobile environment poses great challenges for providing and consuming m-services. Mobile devices have low CPU and memory capacities, limited power supply, small screen size, and restricted input mechanisms. Wireless networks are limited by their small bandwidth. They also suffer from link outages, which result in temporary unavailability. These limitations hinder existing web service technologies from directly working with m-services. For instance, the limited bandwidth may not be enough to convey SOAP messages [21]. In addition, SOAP is expensive for mobile hosts in terms of both power consumption and waiting time. In mobile environment, users' context, such as location and activity, may change rapidly. M-services need to track these changes and provide *context-aware* functionalities. However, service description techniques, such as WSDL, have not provided support to model context. UDDI enables service discovery in the web service framework. However, the multiple costly round trips required by UDDI lookup are troublesome for m-service discovery [21]. The frequent unavailability of wireless network may cause failures in service discovery processes.

7 Conclusion

We describe a service-centric DG framework, WebSenior, to provide services to senior citizens. We implemented actual social services using web services as proxies. In this context, WebSenior is a WSMS that manages the life cycle of government social services. We have shown a set of key service components in WebSenior, consisting of service composition, optimization, and privacy preservation. In particular, WebSenior provides mechanisms that dynamically compose services, select service providers based on their quality attributes, and enforce the privacy of citizens when requesting and receiving government services. One of our ongoing work is to perform extensive experiments to assess the performance the proposed WSMS and its key components.

References

[1] J.L. Ambite et al., "Data Integration and Access," *Advances in Digital Government: Technology, Human Factors, and Policy,* A.K. Elmagarmid and W.J. McIver, eds., pp. 85-106, Kluwer Academic Publishers, 2002.

[2] BEA, IBM, and Microsoft, *Business Process Execution Language for Web Services (BPEL4WS),* http://xml.coverpages.org/bpel4ws.html, 2003.

[3] A. Bouguettaya, A. Rezgui, B. Medjahed, and M. Ouzzani, "Internet Computing Support for Digital Government," *Practical Handbook of Internet Computing,* Chapman & Hall/CRC, 2004.

[4] F. Casati, S. Ilnicki, L. Jin, V. Krishnamoorthy, and M.-C. Shan, "Adaptive and Dynamic Service Composition in eFlow," *Proc. Int'l Conf. Advanced Information Systems Eng. (CAiSE),* June 2000.

[5] S.A. Chun, V. Atluri, and N.R. Adam., "Dynamic Composition of Work-Flows for Customized Egovernment Service Delivery," *Proc. Third NSF Conf. Digital Govt. Research,* pp. 383-389, 2002.

[6] D. Wu et al., "Automating DAML-S Web Service Composition Using Shop2," *Proc. Int'l Semantic Web Conf.,* pp. 195-210, 2003.

[7] S.D. Gribble et al., "Scalable, Distributed Data Structures for Internet Service Construction," *Proc. Fourth Symp. Operating Systems Design and Implementation,* pp. 319-332, 2000.

[8] U. Fayyad, "Data Mining and Knowledge Discovery in Databases: Implications for Scientific Databases," *Proc. Ninth Int'l Conf. Scientific and Statistical Database Management,* pp. 2-11, 1997.

[9] J.M. Hellerstein and M. Stonebraker, "Predicate Migration: Optimizing Queries with Expensive Predicates," *Proc. SIGMOD,* pp. 267-276, 1993.

[10] IBM, *Web Services Flow Language (WSFL),* http://xml.coverpages.org/wsfl.html, 2003.

[11] X. Liu and A. Bouguettaya, "Managing Top-Down Changes in Service-Oriented Enterprises," *Proc. IEEE Int'l Conf. Web Services (ICWS),* pp. 1072-1079, 2007.

[12] Z. Malik and A. Bouguettaya, "Reputation Bootstrapping for Trust Establishment among Web Services," *IEEE Internet Computing,* vol. 13, no. 1, pp. 40-47, Jan./Feb. 2009.

[13] W.C. Mann, "The Aging Population and Its Needs," *IEEE Pervasive Computing,* vol. 3, no. 2, pp. 12-14, Apr.-June 2004.

[14] G. Marchionini, H. Samet, and L. Brandt, "Introduction to Special Issue of Digital Government," *Comm. ACM,* vol. 46, no. 1, pp. 24-27, 2003.

[15] B. Medjahed, B. Benatallah, A. Bouguettaya, A.H.H. Ngu, and A.K. Elmagarmid, "Business-to-Business Interactions: Issues and Enabling Technologies," *VLDB J.,* vol. 12, no. 1, pp. 59-85, 2003.

[16] B. Medjahed and A. Bouguettaya, "A Multilevel Composability Model for Semantic Web Services," *IEEE Trans. Knowledge and Data Eng.,* vol. 17, no. 7, pp. 954-968, July 2005.

[17] Microsoft, *Web Services for Business Process Design (XLANG),* http://xml.coverpages.org/xlang.html, 2003.

[18] OASIS, Web Services Eventing (WS-Eventing), http://www.w3.org/Submission/WS-Eventing, 2006.

[19] M. Ouzzani and B. Bouguettaya, "Efficient Access to Web Services," *IEEE Internet Computing,* vol. 8, no. 2, pp. 34-44, Mar./Apr. 2004.

[20] M.P. Papazoglou and D. Georgakopoulos, "Introduction to Special Issue of Service-Oriented Computing," *Comm. ACM,* vol. 46, no. 10, pp. 24-28, 2003.

[21] T. Pilioura, A. Tsalgatidou, and S. Hadjiefthymiades, "Scenarios of Using Web Services in M-Commerce," *SIGecom Exchanges,* vol. 3, no. 4, pp. 28-36, 2003.

[22] A. Rezgui, M. Ouzzani, A. Bouguettaya, and B. Medjahed, "Preserving Privacy in Web Services," *Proc. Fourth ACM Workshop Information and Data Management (WIDM '02),* pp. 56-62, Nov. 2002.

[23] H. Schuster, D. Georgakopoulos, A. Cichocki, and D. Baker, "Modeling and Composing Service-Based and Reference Process-Based Multi-Enterprise Processes," *Proc. Int'l Conf. Advanced Information Systems Eng.,* pp. 247-263, June 2000.

[24] U. Srivastava, J. Widom, K. Munagala, and R. Motwani, "Query Optimization over Web Services," *Proc. Int'l Conf. Very Large Data Bases (VLDB),* 2006.

[25] S. Tsur, S. Abiteboul, R. Agrawal, U. Dayal, J. Klein, and G. Weikum, "Are Web Services the Next Revolution in e-Commerce? (Panel)," *Proc. Int'l Conf. Very Large Data Bases (VLDB),* 2001.

[26] V. Venkatesh, V. Ramesh, and A.P. Massey, "Understanding Usability in Mobile Commerce," *Comm. ACM,* vol. 46, no. 12, pp. 53-56, 2003.

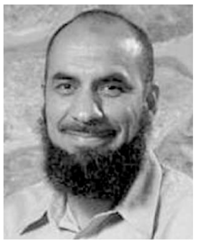
[27] W3C, Web Services Notification (WSN), http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsn, 2006.

[28] X. Yang, A. Bouguettaya, B. Medjahed, H. Long, and W. He, "Organizing and Accessing Web Services on Air," *IEEE Trans. Systems, Man, and Cybernetics—Part A: Systems and Humans,* vol. 33, no. 6, pp. 742-757, Nov. 2003.

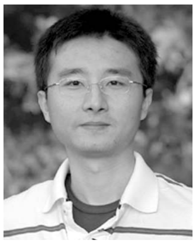
[29] Q. Yu and A. Bouguettaya, "Framework for Web Service Query Algebra and Optimization," *ACM Trans. Web,* vol. 2, no. 1, 2008.

[30] Q. Yu, X. Liu, A. Bouguettaya, and B. Medjahed, "Deploying and Managing Web Services: Issues, Solutions, and Directions," *VLDB J.,* vol. 17, no. 3, pp. 537-572, 2008.

[31] L. Zeng, B. Benatallah, M. Dumas, J. Kalagnanam, and Q. Sheng, "Quality-Driven Web Service Composition," *Proc. Int'l Conf. World Wide Web (WWW),* 2003.



Athman Bouguettaya received the PhD degree in computer science from the University of Colorado at Boulder in 1992. He is Head of School, Computer Science and Information Technology at RMIT, Melbourne, Australia. He was a science leader at the CSIRO ICT Centre, Canberra, Australia. He was also previously a tenured faculty member in the Computer Science Department at the Virginia Polytechnic Institute and State University (commonly known as Virginia Tech). He currently holds adjunct professorships at the Australian National University, Canberra, the University of Queensland, Brisbane, Australia, and Macquarie University, Sydney, Australia. He is on the editorial boards of several journals, including the *VLDB Journal*, the *Distributed and Parallel Databases Journal*, the *International Journal of Cooperative Information Systems*, and the *IEEE Transactions on Services Computing*. He guest edited the *IEEE Internet Computing* special issue on database technology on the web and the *ACM Transactions on Internet* special issue on Semantic Web services. He served as the program chair of the 2008 International Conference on Service Oriented Computing (ICSOC 2008), the 20th Australasian Database Conference (ADC 2009), and the IEEE RIDE Workshop on Web Services for E-Commerce and E-Government (RIDE-WS-ECEG 2004). He has published more than 130 articles in journals and conferences in the area of databases and service computing (e.g., *IEEE Transactions on Knowledge and Data Engineering*, *ACM Transactions on the Web*, the *International Journal on Very Large Data Bases*, SIGMOD, ICDE, VLDB, and EDBT). His current research interests are in the foundations of web service management systems. He is a fellow of the IEEE and the IEEE Computer Society and a senior member of the ACM.



Qi Yu received the PhD degree in computer science from the Virginia Polytechnic Institute and State University (Virginia Tech). He is an assistant professor in the College of Computing and Information Sciences, Rochester Institute of Technology. His current research interests lie in services computing and data management and mining. His publications have mainly appeared in well-known journals (e.g., the *International Journal on Very Large Data Bases* and *ACM Transactions on the Web*) and conference proceedings (e.g., ICWS). He serves as a program committee member on several conferences, including the International Conference on Service Oriented Computing (ICSOC 2008), the International Conference on Information Reuse and Integration (IRI 2009, 2010), and the Australasian Database Conference (ADC 2009, 2010). He is also a reviewer for various journals, including the *VLDB Journal*, *ACM Transactions on the Web*, the *IEEE Transactions on Services Computing*, and the *International Journal on Distributed and Parallel Databases*. He is a member of the IEEE.

Xumin Liu received the PhD degree in computer science from the Virginia Polytechnic Institute and State University (Virginia Tech). She is an assistant professor in the Department of Computer Science, Rochester Institute of Technology. Her research interests lie in the general field of data management and service computing with special focuses on change management and service composition. Her research has been published in various international journals and conferences, such as the *International Journal on Very Large Data Bases*, the IEEE International Conference on Web Services (ICWS), and the International Conference on Collaborative Computing: Networking, Applications, and Worksharing (CollaborateCom). She frequently serves as a program committee member or a reviewer for various international conferences. She is also a reviewer for the *IEEE Transactions on Services Computing* and the *International Journal on Distributed and Parallel Databases*. She is a member of the IEEE.

Zaki Malik received the PhD degree in computer science from the Virginia Polytechnic Institute and State University (Virginia Tech) in 2008. He is currently an assistant professor of computer science at Wayne State University, and the director of the Services Computing Research Laboratory. His research interests include trust management, digital government, semantic integration, and services computing. He has published several papers in top-tier journals (such as the *International Journal on Very Large Data Bases* and the *World Wide Web Journal*) and conferences (such as ICWS and ICSOC) in the above areas. He also serves on program committees of database and services-oriented computing conferences, and reviews for journals. He is a member of the IEEE.