

# WebDG – A Platform for E-Government Web Services

Athman Bouguettaya<sup>1</sup>, Brahim Medjahed<sup>2</sup>, Abdelmounaam Rezgui<sup>1</sup>,  
Mourad Ouzzani<sup>3</sup>, Xumin Liu<sup>1</sup>, and Qi Yu<sup>1</sup>

<sup>1</sup> Department of Computer Science, Virginia Tech  
{athman, rezgui, xumin1, qyu}@vt.edu

<sup>2</sup> Dept of Computer and Information Science, University of Michigan, Dearborn  
brahim@umd.umich.edu

<sup>3</sup> Department of Computer Sciences, Purdue University  
mourad@cs.purdue.edu

**Abstract.** Web services are deemed as the natural choice for deploying e-government applications. Their use enables e-government to fully get advantage of the envisioned *Semantic Web*. In this paper, we propose WebDG, a comprehensive *Web Service Management System* for e-government applications. It aims to improve government-citizen interactions through an infrastructure built around the “life experience” of citizens. WebDG provides a framework for automatically composing e-government services, optimized querying services, and preserving privacy.

## 1 Introduction

*Digital government* applications are becoming commonplace. A major propelling technology for e-government is the emerging concept of *Web services*. The nexus between the two is becoming very strong as Web services provide the platform of choice for deploying the different functionalities offered by governments and supporting interactions with both government and non-government applications. A *Web service* is a set of related functionalities that can be programmatically accessed and manipulated through the Web [16]. Examples of e-government Web services include electronic tax filing, department of motor vehicle driver’s license service, and social services (e.g., health insurance for disadvantaged people). The powerful concept of Web service is taking root because of the convergence of government and business efforts to make the Web the place of choice for all types of human activities [9, 15].

The ability to efficiently access and share e-government services is a critical step towards the full deployment of digital government. This requires the development of techniques to address various challenging issues. Required techniques include service description, discovery, querying, composition, monitoring, security, and privacy. All these techniques would be part of a comprehensive *middleware* for managing *autonomous* and *heterogeneous* Web services. For that purpose, we are investigating the architectural components of a *Web Services Management System* (*WSMS*). The aim of a *WSMS* is to do for Web services

what DBMSs have done for data. Users will no longer need to think in terms of *data* but rather *services*. Web services will be treated as *first-class* objects that can be manipulated as if they were pieces of data.

In this paper, we present a comprehensive WSMS for e-government called *Web Digital Government* (WebDG). WebDG embraces emerging standards for describing (WSDL), discovering (UDDI), and invoking (SOAP) Web services [6]. Adopting Web services in WebDG enables: (i) *standardized* description, discovery, and invocation of welfare applications, (ii) *composition* of pre-existing services to provide *value added* services, and (iii) *uniform* handling of privacy. WebDG is built around a collection of features that include a framework for composing e-government services, optimized querying services, and preserving privacy.

The remainder of this paper is organized as follows. Section 2 gives a scenario of e-government applications. It shows the drawbacks of current system and what WebDG aims to do. Section 3 examines three features of WebDG, including composing e-government services, optimized querying services, and preserving privacy. Section 4 then describes the implementation of WebDG. Section 5 gives our concluding remarks.

## 2 A Scenario of E-Government Applications

To illustrate the drawbacks of the current system and how WebDG can help, we examine a typical scenario in this section. One of the major concerns of e-government is to improve government-citizen interactions using information and communication technologies [9]. In our e-government project, we have teamed up with *Indiana's* FSSA. Collecting social benefits is currently a frustrating and cumbersome task in FSSA. Citizens must often visit different offices located within and outside their hometown. Additionally, case officers must delve into a wealth of proprietary applications to access welfare programs that best meet citizens' needs.

Let us consider the following scenario typical to FSSA application domain. A pregnant teen, let's call her Mary, goes to an FSSA office to collect social benefits. Mary needs a government-funded health insurance program. She would also like to receive nutritional advice. Because Mary will not be able to take care of the future newborn, she is also interested in finding a foster family.

The case officer, let's call him John, must access all relative services among various agencies to fulfill Mary's needs. He must first manually look up which social programs offer health insurance and food assistance for Mary. He then needs to find a foster family service. For each service, John needs to select an appropriate provider. The choice of the provider is mostly based on John's expertise and some information gathered through different means (e.g., Web sites, brochures). Since there might be a large number of candidate providers, choosing one that best fits Mary's requirement is by no means an easy task. Finally, John chooses Medicaid (a healthcare program for low-income citizens) and WIC (a federally funded food program for Women, Infants, and Children). Assuming

Medicaid is locally accessible, John can connect to the corresponding application and interact with it. Because WIC is a federal program, however, John has no direct access to the corresponding application. That means Mary must visit another agency, perhaps in a different town, to apply for the benefit.

Still more difficulties arise when John tries to find a foster family service. Using local resources, he finds no matching program, although Teen Outreach Pregnancy (TOP), an FSSA partner, does offer such services. To complicate things further, each time John connects to an application, he has to make sure that it abides by privacy rules related to the access to and use of sensitive information such as Mary's social security number (SSN).

This process of manually researching and accessing individual services is clearly time-consuming. It would be more efficient if John could specify Mary's needs once and address them all together. He could then seamlessly access all related services through a single access point. In addition, John manually selects a service or the combination of several services for Mary from a large number of candidate providers. This manual selection can hardly guarantee optimal outcomes. John needs a service query mechanism to help him efficiently select the best services. Moreover, during the process of applying services, Mary's privacy may be released because some providers may need such information to offer services. WebDG provides solutions for all above issues and offers comprehensive support for government-citizen interactions.

### 3 WebDG: A WSMS for Digital Government

WebDG is a WSMS for digital government. It provides a framework for efficiently accessing e-government services while preserving citizens' privacy. Its main contributions revolve around three features, including composing e-government services, optimized querying e-government services, and preserving privacy. We overview WebDG's approach for these features in this section.

#### 3.1 Composing E-Government Services

We propose a new approach for the (*semi*) *automatic composition* of Web services. Automatic composition is expected to play a major role in enabling the envisioned *Semantic Web* [4]. WebDG's approach for service composition is particularly suitable for e-government applications. It focuses on a select type of users (citizens and case officers) to provide customized e-government services. Composing services in WebDG includes four phases: *specification*, *matchmaking*, *selection*, and *generation*. In the following, we focus on the *matchmaking* phase. The other phases are outlined for the sake of completeness.

**Specification.** Service composers define high level descriptions of the desired composition via an XML-based language called CSSL (*Composite Service Specification Language*). They simply provide abstract definitions of the operations to be performed without referring to existing Web services. CSSL uses a subset of WSDL service interface elements and extends it to allow the: (1) description

of semantic features of Web services and (2) specification of the control flow between composite services operations. Defining a WSDL-like language makes the definition of composite services as simple as the definition of simple (i.e., non composite) services. It also allows the support of recursive composition.

**Matchmaking.** Once CSSL specifications are provided, the next step is to generate corresponding composition plans. The matchmaking phase includes two issues: how to choose e-government services to generate plans and how to ensure composabilities of these services. In the following, we propose our approach to address these two issues.

The matchmaking phase automatically generates *composition plans* that conform to users' specification. A *composition plan* refers to the list of outsourced services and the way they interact with each other (plugging operations, mapping messages, etc). To accelerate the discovery of component services, we organize WebDG services into *communities* [8, 5]. Communities provide means for an ontological organization of the available service space based on *categories*. All services that have similar category belong to the same community. We define an ontology for e-government service called *Category*. We assume that government social agencies would agree on the ontology ahead of time. The *Category* ontology contains four attributes: *name*, *synonyms*, *specialization*, and *operations*. The *name* gives the domain of interest of the current community (e.g., "healthcare"). The *synonyms* attribute contains a set of alternative names. For example, "medical" is a synonym of "healthcare". *Specialization* is a set of characteristics of the current category. For example, "insurance" and "children" are specializations of the "healthcare" category. The *Operations* attribute gives a list of generic operations provided by community services. Each operation has a set of *input* and/or *output* parameters. Each parameter has an XML Schema data type. An operation also has a *Type* element that belongs to an ontology *Type*. This ontology includes *name*, *synonyms*, and *specialization* attributes. The *name* gives the business functionality offered by the current operation (e.g., "eligibility", "counseling"). *Synonyms* and *specialization* attributes are defined as in *Category*.

Providers (e.g., FSSA bureaus) identify the community of interest and register their services with it. Services can leave and reenter a community at any time during their life-span. During the registration process, providers must define the mappings between generic operations defined in their community and those defined in their service. A service may offer all or some of the operations defined within a community. For each generic operation, it may use all operation's parameters, a subset of those parameters, and/or add new parameters.

A major issue addressed by WebDG's matchmaking algorithm is *composability* of the outsourced services [4]. We propose a set of rules to check composability for e-government services. These include *operation semantics* composability that compares the semantics of service operations and *composition soundness* that checks whether combining Web services in a specific way is worthwhile. We first give the definition of *e-government services* to formally describe the set of composability rules.

**Definition 1** - *E-government Service*. An *e-government service*  $ES_i$  is defined by a tuple  $(Description_i, OP_i, Bindings_i, Purpose_i, Category_i)$  where:

- $Description_i$  is a text summary about the service features.
- $OP_i$  is a set of operations provided by  $ES_i$ .
- $Bindings_i$  is the set of binding protocols supported  $ES_i$ .
- $Purpose_i = \{Purpose_{ik}(op_{ik}) \mid op_{ik} \in OP_i\}$  is a set of  $ES_i$  operations' purposes.
- $Category_i = \{Category_{ik}(op_{ik}) \mid op_{ik} \in OP_i\} \cup \{Category_i(ES_i)\}$  is a set of  $ES_i$  operations' categories.  $\diamond$

Operation semantic composability compares the categories or domains of interest (e.g., “healthcare”, “adoption”) of each pair of interacting operations. It also compares their purposes or functionalities (e.g., “eligibility”, “counseling”). To define *compatibility* between operation categories, let us consider the two operations  $op_{ik} = (D_{ik}, M_{ik}, In_{ik}, Out_{ik}, P_{ik}, C_{ik}, Q_{ik})$  and  $op_{jl} = (D_{jl}, M_{jl}, In_{jl}, Out_{jl}, P_{jl}, C_{jl}, Q_{jl})$ . We say that  $C_{ik}$  is *compatible with*  $C_{jl}$  if:

1.  $(C_{ik}.Domain = C_{jl}.Domain)$  or  $(C_{ik}.Domain \in C_{jl}.Synonyms)$  or  $(C_{jl}.Domain \in C_{ik}.Synonyms)$  or  $(C_{ik}.Synonyms \cap C_{jl}.Synonyms \neq \emptyset)$ ; and
2.  $C_{ik}.Specialization \subseteq C_{jl}.Specialization$

**Definition 2** - *Operation Semantics Composability*. We say that  $op_{ik} = (D_{ik}, M_{ik}, In_{ik}, Out_{ik}, P_{ik}, C_{ik}, Q_{ik})$  is *operation semantics composable* with  $op_{jl} = (D_{jl}, M_{jl}, In_{jl}, Out_{jl}, P_{jl}, C_{jl}, Q_{jl})$  if (i)  $P_{ik}$  is *compatible with*  $P_{jl}$  and (ii)  $C_{ik}$  is *compatible with*  $C_{jl}$ .  $\diamond$

*Composition soundness* checks whether combining a set of services in a specific way provides an added value. For that purpose, we introduce the notion of *composition template*. A *composition template* is built for each composition plan generated by WebDG. It gives the general structure of that plan. We also define a subclass of templates called *stored templates*. These are defined *a priori* by government agencies. Since stored templates inherently provide added values, they are used to test the *soundness* of composition plans.

**Definition 3** - *Composition Soundness*. A composition of services is *sound* if its template is a subgraph of a *stored template*.  $\diamond$

**Selection.** At the end of the matchmaking phase, several composition plans may have been generated. To facilitate the *selection* of relevant plans, we define *Quality of Composition (QoC)* parameters. Examples of such parameters include time, cost, and plan's ranking. Composers define (as part of their profiles) thresholds corresponding to *QoC* parameters. Composition plans are returned only if the values of their *QoC* parameters are greater than their respective thresholds.

**Generation.** This phase aims at *generating* a detailed description of a composite service given a selected plan. This description includes the list of outsourced services, mappings between composite service and component service operations, mappings between messages and parameters, and flow of control and data between component services. Composite services are generated in emerging standards for service composition such as BPEL4WS [3], WSFL [7], and XLANG [2].

### 3.2 Optimized Querying of E-Government Services

WebDG provides a *query scheme* that offers *database-like* query facilities over Web services [11]. Users submit queries that are answered through a combined access to various Web services. The challenge is then to devise the “best” alternative of Web services combinations with respect to the delivered quality. Fundamental premises of the querying scheme is that Web services are *a priori* unknown, their number is potentially very large, and they are usually autonomous, heterogeneous, and highly volatile [13]. We propose an *optimization model* based on *Quality of Service (QoS)* that would capture users’ requirements for efficiency.

**E-Government Service Querying.** A fundamental challenge in enabling e-government service queries is how to obtain the combination of actual operations from the declarative expression of a query. For that purpose, WebDG contains a three-level query paradigm where queries go through several transformations that lead to the *service execution plan* [10, 12]. The query paradigm includes *query level*, *virtual level*, and *concrete level*. Query level allows users to submit database-like queries. Each relation defined at the query level is mapped to one or a set of *virtual operations* at the virtual level. A virtual operation contains elements of input variables, output variables, category, and function description. The virtual operations are mapped to concrete e-government services at the concrete level.

In the virtual operations matching phase, it is not always possible to find an exact match for a given virtual operation. In addition, the same functionality may be offered in various ways by different e-government services. Users may be inclined to accept similar or close answers to their queries. This is especially true in the context of social services where the objective is to get whatever social benefits are available for a needy citizen. Consequently, we propose a multi-level model for virtual operation matching. This model includes *exact match*, *overlap match*, and *partial match*. These three levels reflect different *matching degree* that quantifies how exact the matching is. This would help the citizen and the case manager in assessing the results of their queries. The matching degree has a direct impact on the quality of the query results and subsequently on the optimization process.

**Optimization Model.** Given a query, an important challenge for the system is to find the “best” query execution plan with respect to an objective function. Indeed, the resolution of any query may lead to various alternatives with disparate qualities. Quality of service (*QoS*) is playing a crucial role in assessing the added-value of competing e-government services [17]. In our approach, e-government services are selected and combined based on the *QoS* they offer adjusted through a *dynamic rating scheme* and *multilevel matching*. Each time an e-government service is selected in solving a query, it is rated by comparing its advertised *QoS* with its actual *QoS*. In addition, different *levels of matching* have been considered in matching virtual and concrete operations. Each level has a *matching degree* that is also used to adjust the objective function.

*QoS* is defined through a number of parameters supplied by the service providers. The objective of the optimization process is to maximize or minimize each value. In the proposed system, we consider the *QoS* parameters including *latency*, *availability*, and *security*. Latency represents the average time it takes for an operation to return results after its invocation. Availability defines whether the e-government service is present and ready to be invoked. It represents the probability that a service is available. Security reflects the ability to provide confidentiality and non-repudiation of exchanged information. This is crucial for digital government applications that indeed manipulates large amounts of sensitive information. It is clear that these are not the only parameters that may be used to assess the quality of e-government services. Other parameters include accessibility, reliability, etc.

Due to the different fluctuations that may occur with an e-government service, the *QoS* advertised by that e-government service may not be always fulfilled. Furthermore, the e-government service may change some of its *QoS* parameter values over time. To ensure that *QoS* parameters are used in a way that represents the *actual* quality of an e-government service, we propose to adjust the advertised values of those parameters. The idea is to *rate* e-government services by monitoring them and computing the *delivered QoS*. The *promised QoS* (*pQoS*) is the value of the *QoS* parameters advertised by the service provider. The *delivered QoS* (*dQoS*) is the value of the *QoS* parameters obtained by monitoring the Web service.

Any new e-government service receives initially the highest rating. Rates range over a  $[0, 100]$  scale where 100 is the highest value. E-government services with a negative *QoS distance* above a certain negative threshold will have their rating lowered. On the contrary, if the *QoS distance* has a positive value greater than a certain positive threshold, the Web service rating is increased if it does not have already the highest value. In subsequent queries, *QoS* parameters are weighted by the ratings of the corresponding e-government services.

### 3.3 Preserving Privacy

Privacy is a major issue that e-government needs to address [17]. Citizens generally must divulge sensitive information, such as their SSN or salary, to access e-government services. Two characteristics add to the complexity of the privacy problem in e-government: sharing of citizens information among government agencies and citizens differing privacy requirements. Privacy is generally misperceived as an issue whose natural solution consists of good security mechanisms. Security and privacy are tightly interrelated issues, but secure e-government infrastructures do not necessarily ensure privacy. Our system focuses on privacy enforcement; we assume that appropriate security mechanisms, such as secure communication channels, already exist within the e-government environment.

**Privacy Model.** A typical citizen-government interaction involves three participants: users, services, and databases. This naturally defines a three-layered model for privacy [14].

The first layer of the *privacy model* is *user privacy*. Users of an e-government service include persons (e.g., citizens and case officers), applications, and other e-government services. In many cases, users interacting with an e-government service are required to provide a significant amount of personal sensitive information (e.g., their SSN, credit card number and address). Users of e-government services, however, may expect or require different levels of privacy according to their perception of the *information sensitivity*. The user's perception of privacy also depends on the *information receiver* and the *information usage* [1]. The set of privacy preferences applicable to a user's information is called *user privacy profile*. A user privacy profile is typically defined by the user but can also be uniformly set for a group of individuals. Privacy profiles are *dynamic*: users can create, view, update, or delete their privacy profiles. To provide support for resolving legal disputes over privacy violation, the underlying Web service architecture must trace all of these operations. We also define a user's *privacy credentials* as a signature that is typically appended to any request that the user submits to the Web service. They determine the *privacy scope* for the corresponding user. A privacy scope for a given user defines the information that an e-government service can disclose to that user.

The second layer of the *privacy model* is *service privacy*. An e-government service generally has its own *privacy policy* that specifies a set of rules applicable to *all* users. Service privacy generally specifies three types of policy: *usage* policy, *storage* policy, and *disclosure* policy. The *usage* policy states the purposes for which the information collected can be used. For example, consider an e-government service **Medicaid** that provides healthcare coverage for low-income citizens. **Medicaid** may state that the information collected from citizens will not be used for purposes other than those directly related to providing health services to citizens. The *storage* policy specifies whether and until when the information collected can be stored by the service. For example, **Medicaid** may state that the information it collects from citizens will remain stored in the underlying databases one year after they leave the welfare program. The *disclosure* policy states if and to whom the information collected from a given user can be revealed. This information may relate to individual persons or to *groups* of individuals. For example, the privacy policy of the service **Medicaid** may state that external users cannot access statistical information that reveals general characteristics of the beneficiaries (e.g., average income, racial background distribution, etc).

The third layer of the *privacy model* is *data privacy*. A data object may be accessed by several e-government services. For example, consider the US National Database for New Hires (NDNH) that contains information about over 200 millions hired employees. A record in this database can be accessed (using an e-government service) by an IRS officer to check the accuracy of an employee's tax form. It may also be accessed (using another e-government service) by an officer at a child support agency to check whether a parent is compliant with his child support obligations. This shows that different e-government services may need different information from the same data object. Thus, data objects must

be able to *expose* different views to different e-government services. For each data object, we define a *data privacy profile* that specifies the access views that it exposes to the different e-government services.

**Privacy Enforcement.** An important premise in our privacy enforcement approach is that users access databases through e-government services. When a service receives a request from a given user, it first checks that the user has the necessary credentials to access the requested operation according to its privacy policy. Lets look at the Medicaid service, which states that the only person who can update a citizens privacy profile is that citizen. If the request is valid, the service translates the users request into an equivalent data query and submits it to the appropriate government database management system.

Before the Medicaid service submits a query to the DBMS, it sends the query through a privacy preserving data filter (DFilter). This 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 to enforce all the privacy constraints. For example, the QRM deletes the salary field from a service request that translates into the SQL query `select name, age, salary from Medicaid.enrollees` before submitting it to the DBMS if the credential of the user who made the request does not allow access to enrollees salaries.

The privacy profile manager (PPM) enforces privacy at a finer granularity than the CCM does. The local CCM might decide that an organization can access local information regarding health records for a group of citizens, but some of those citizens might explicitly request that parts of their records not be made available to third-party entities. In this case, the local PPM would discard those parts from the generated result. The PPM is a translation of the consent-based privacy model in that it implements individual citizens privacy preferences. It maintains a repository of privacy profiles that stores individual privacy preferences. The PPM also handles citizens requests for updating their privacy profiles.

## 4 Implementation

Figure 1 shows the WebDG system as implemented across a network of *Solaris* workstations. Citizens and case officers access the system via a graphical user interface implemented in HTML and Java servlets. WebDG currently includes seven FSSA applications implemented in Java (JDK 1.3). The Axis Java2WSDL utility in *IBMs Web Services Toolkit* automatically generates WSDL descriptions from Java class files, which WebDG publishes into a UDDI registry. WebDG uses the service management client within Apache SOAP 2.2 to deploy e-government services. Apache SOAP provides a server-side infrastructure for deploying and managing services, and a client-side API for invoking those services. Each service

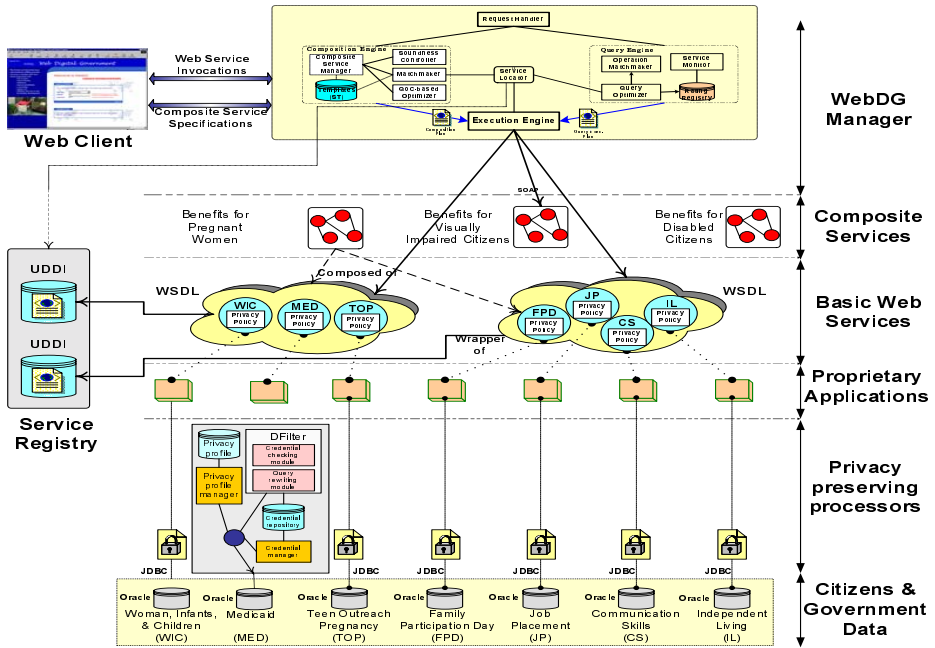


Fig. 1. WebDG Architecture

has a deployment descriptor that includes the unique identifier of the Java class to be invoked, the session scope of the class, and operations in the class available for the clients. WebDG deploys each service using its descriptor and the URL of the Apache SOAP servlet `rpcrouter` as input arguments.

The *WebDG manager* is at the system’s core. The *service locator* (SL) looks up WSDL descriptions in the registry. Once the *execution engine* discovers a service, it invokes the services operations through a SOAP binding stub.

The *composite service manager* (CSM) uses the Java API for XML Processing (JAXP) to parse the XML-based composite service specifications. It then returns them to the *matchmaker*, which checks composability rules. The matchmaker then sends each composite service operations category to the SL. The SL parses each located services WSDL description and returns it to the matchmaker. The SL retrieves only services whose category is compatible with the operations category. After checking composability, the matchmaker generates composition plans and sends them to the *optimizer*, which selects plans based on quality of composition (QoC) parameters. Users define thresholds for such parameters as time, cost, and the plans relevance to the users specification. The optimizer returns plans to the matchmaker if the QoC parameter values are greater than the users thresholds. The matchmaker forwards the selected plans to the *soundness controller* (SC) to check that the way they are combined provides added value. The SC then returns plans, along with their compatible stored templates (if



Fig. 2. WebDG Interface for Preserving Privacy

any), to the CSM. The CSM then sends the plans to the *execution engine*. The *execution engine* enacts the plans by actually invoking e-government services using SOAP. We use *SOAP Binding Stubs* which are implemented using Apache SOAP API for this purpose.

The *service query engine* (SQE) is responsible for the correct and optimal execution of e-government service queries in WebDG. The *operation Matchmaker* (OM) interacts with the SL to retrieve the services' descriptions in WSDL and determines the concrete operations to use in the service execution plan. WSDL descriptions (augmented with semantic attributes that we have defined) are parsed and concrete operations are matched to virtual operations using one of the matching modes. The *monitoring agent* (MA) monitors e-government service invocations. Its goal is to assess their behavior in terms of the delivered *QoWS*. The monitoring agent maintains a local repository for ratings and other information to compute those ratings. The *query optimizer* (QO) is the central component of the SQE. It determines the best service execution plan for a given query based on the optimization model that we presented in Section 3.2. After the optimizer generates an efficient service execution plan, the plan is handed over to the execution engine.

When service operations attempt to access the FSSA databases, a privacy-preserving processor intercepts the operation invocations and allows or disallows access based on privacy profiles, privacy credentials, and data filters. Fig-

ure 2 shows TOP's Search Family Adoption operation, which returns information about foster families in a given state, in this case Virginia. The value "no right" for the attribute **Race** means that the current user does not have system permission to access information about family F1's race. The value "not accessible" for the **Household Income** attribute indicates that family F1 does not want to disclose that sensitive information.

## 5 Conclusion

In this paper, we describe a comprehensive WSMS for digital government, WebDG, for the efficient deployment of e-government services. WebDG uses Web services as an effective vehicle to provide government services to those citizens who need the help the most. Web services empower e-government with features from the envisioned Semantic Web. The main contributions of WebDG revolve around three features: composing e-government services, optimized querying of e-government services, and preserving privacy. We describe a Web service based implementation of the WebDG system to deploy several FSSA applications as e-government services.

## References

1. A. Adams. User's Perception of Privacy in Multimedia Environment. *PhD thesis, School of Psychology, University College London*, 2001.
2. S. Aissi, P. Malu, and K. Srinivasan. E-Business Process Modeling: The Next Big Step. *IEEE Computer*, 35(5), May 2002.
3. BEA, Microsoft, and IBM. *Business Process Execution Language for Web Services (BPEL4WS)*. <http://xml.coverpages.org/bpel4ws.html>, 2003.
4. T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, May 2001.
5. Bouguettaya, A. Elmagarmid, B. Medjahed, and M. Ouzzani. Ontology-based support for digital government. In *VLDB 2001*, Roma, Italy, September 2001.
6. F. Curbera, M. Duftler, R. Khalaf, W. Nagy, N. Mukhi, and S. Weerawarana. Unraveling the Web Services Web: An Introduction to SOAP, WSDL, and UDDI. *IEEE Internet Computing*, 6(2), February 2002.
7. IBM. *Web Services Flow Language (WSFL)*. <http://xml.coverpages.org/wsfl.html>, 2003.
8. B. Medjahed, A. Bouguettaya, and A. Elmagarmid. Composing Web Services on the Semantic Web. *The VLDB Journal, Special Issue on the Semantic Web*, September 2003 (to appear).
9. B. Medjahed, A. Rezgui, A. Bouguettaya, and M. Ouzzani. Infrastructure for E-Government Web Services. *IEEE Internet Computing*, 7(1), January 2003.
10. M. Ouzzani and A. Bouguettaya. A query paradigm for web services. In *1st International Conference on Web Services*, Las Vegas, NV, USA, June 2003.
11. M. Ouzzani, A. Bouguettaya, and B. Medjahed. Optimized querying of e-government services. In *The dg.o 2003 NSF Conference for Digital Government Research*, Boston, USA, May 2003.

12. M. Ouzzani and B. Bouguettaya. Efficient access to web services. *IEEE Internet Computing*, 37(3), March 2004.
13. M. Ouzzani and B. Bouguettaya. Query processing and optimization on the web. *Distributed and Parallel Databases, an International Journal*, 15(3), May 2004.
14. A. Rezgoui, M. Ouzzani, A. Bouguettaya, and B. Medjahed. Preserving Privacy in Web Services. In *the 4th International ACM Workshop on Web Information and Data Management*, November 2002.
15. Stanley Y.W. Su, Herman Lam, Minsoo Lee, Sherman Bai, and Zuo-Jun Shen. An information infrastructure and e-services for supporting Internet-based scalable e-business enterprises. In *Enterprise Distributed Object Computing Conference, 2001. EDOC '01. Proceedings. Fifth IEEE International*, pages 2–13, Seattle, WA, USA, September 2001.
16. S. Tsur, S. Abiteboul, R. Agrawal, U. Dayal, J. Klein, and G. Weikum. Are Web Services the Next Revolution in e-Commerce? (Panel). In *Proceedings of 27th International Conference on Very Large Data Bases*, Rome, Italy, September 2001. Morgan Kaufmann.
17. S. Vinoski. Service discovery 101. *IEEE Internet Computing*, 7(1), Jan/Feb 2003.