Use of web service orchestration strategies in operations on digital democracy platform

Carlos González Martínez¹, Emilia Pérez Belleboni², Sergio Sánchez García³, Ana Gómez Oliva⁴ y Jesús Moreno Blázquez⁵

Universidad Politécnica de Madrid. Departamento de Ingeniería y Arquitecturas Telemáticas, Tel: +34913367818, Fax: +34913367817
{cgonzalez1, belleboni2, sergio3, agomez4, moreno5}@diatel.upm.es

Abstract

Platforms of digital democracy, including both those in platforms of e-government and those operating independently, provide a channel of citizen participation enabling democratic decision-making processes. This paper presents a proposal for an e-democracy platform based on web services technology that is configurable and extensible for use in different scenarios of citizen participation. Operations that this platform must provide to applications are provided, and its breakdown in web services, with the use of service orchestration techniques to define plans for controlling the execution of services in the proper order.

Keywords: Orchestration strategies, eDemocracy, web Services

1. Introduction

In recent years, the development of the Internet and the rise of new technologies in the information and telecommunications age have paved the way for the development of a set of applications that have spurred the evolution of technological solutions themselves. Governments and public administrations, which have often promoted this development by financing research projects in this field of technology, have not stood aloof from the development and implementation of applications facilitating the performance of productive processes linked to administrative management. The most well-known example of the latter is e-government. We will generally use this term to refer both to the provision of administrative services through electronic means – which has been called the electronic administration – and the change in administrative processes of government allowing the latter to concentrate efforts on citizens and their needs, rather than on the bureaucratic organization on which this work is grounded. Thus, it involves both a change of technology and a change of mentality.

In recent times, a new term has appeared, rendered as either e-democracy or electronic democracy, a term which is often subsumed under or confused with e-government. Under the guise of e-democracy one can speak of systems to transfer information to users through web technologies, support systems for certain administrative processes such filling tax returns, or of complex, insufficiently defined systems that include the above components combined with other that can strengthen interactive and participatory democracy. In this paper, electronic democracy shall be understood as a service made available to citizens to allow them to freely express their opinions – within the limits of mutual respect – with the basic objective of drawing conclusions for decision-making processes, in a project that aims to foster better relations among citizens and between citizens and the authorities.

In Europe, specifically in the Fourth, Fifth and Sixth Framework Programs of the EU, a series of projects have been undertaken aimed at promoting e-government and decision-making in a context we would call e-democracy. In our study of the most significant projects in this field, namely DEMOS [1], DUNES [2], WEBOCRACY [3], EURO-CITI [4], SMARTGOV [5], EDEN [6], eUSER [7], KEeLAN [8] evaluated in terms of issues such as support for decision-making, conditions in treatment of security and techniques used for development, we have identified opportunities for improvement to be addressed prior to a final deployment of these services and applications.

In the systems analyzed, we have found that they fail to take into account – or they do so only in a limited way – problems that may arise from a lack of security in treatment of information and communications, and they ignore issues as important as user anonymity, confidentiality in the exchange of information, information integrity and non-repudiation. Different experiences exist of integration of security in these systems, such as the CASENET project [9] developed under the Fifth Framework Programme and a series of proposals such as SecureUML [10] and UMLSec [11]. However, these proposals have yet to include the aforementioned security services, and level of maturity of
the tools is still insufficient for the development of applications.

Designing an electronic platform of citizen participation involves two aspects related to requirements that demand special attention: one is user orientation and the other is technology orientation. The former concerns user demands in issues such as security, accessibility, ease of use, cost and so on. The second issue pertains to the deployment of applications and a design strategy that enables configuration for multiple scenarios, while ensuring flexibility and extensibility. In the last of these, little information is available on the development model of the aforementioned projects, but they have generally used J2EE technologies for the generation of proprietary platforms based on web-distributed applications.

The objective of this paper is to present the design of a platform of citizen participation that provides efficient solutions to these problems of security and extensibility. To do so, a service-oriented architecture (SOA) will be used, as this sort of architecture allows for granularity of services so that combinations of the same (orchestration) can form application operations adapted to different scenarios, and they can be easily extensible by adding new services or flexibly adapted to the scenario implemented.

This paper is the result of the research work undertaken in the projects TIC2003-2141 Development of secure telematic platform to support scenarios of digital democracy and TSI2006-4864 Telematic platform of electronic administration based on choreography of services, both financed by the Spanish Ministry of Education and Science as part of the National R+D Plan.

2. Characteristics of e-democracy platform

In this e-democracy platform, a series of applications have been defined that enable users to interact with the system in diverse ways: debates, votes, surveys, wikis, blogs, notice boards and notifications and general handling of the census. All these applications can be configured to support different levels of security and different requirements of participation, making it adaptable to the specific scenario in which the platform is to be used (municipality, company, university, association, etc).

To achieve the desired functionality, a series of requirements have been identified that must be applied in the design of the platform:

- Flexibility in executing operations: the platform must be capable of choosing between different services, with the availability of different families of services, that is, sets of equivalent services that offer a similar functionality but which differ in their interfaces, implementation or location.
- Extensible: The platform must make it easy to include new services and operations.
- Security: The platform must ensure the security of information both in transit and when stored.
- The platform must be built on the basis of stable recommendations or standards issued by international bodies (such as W3C or OASIS) and use, in so far as possible, open source tools in its development.

For all the foregoing, we have chosen a web services architecture: the programming environments are extensible, allowing for the addition of new elements with a relatively small impact; it is a technology which is independent of both the development platform and programming languages, thus allowing for the use of non-proprietary environments; it does not require special user applications; it adapts well to different interfaces, and it uses transport and packaging mechanisms based on Internet specifications (TCP/IP, HTTP, SOAP, etc.)

3. Architectural Model

The architecture proposed for the service integration platform for e-democracy (see Figure 1) can channel and manage all interactions between users and services.

![Figure 1. Platform architecture](image-url)

It is composed of three blocks or functional levels. The first level is the **Access System**: that is, the application(s) through which users interact with the platform to access the services offered:

- **Intrinsic services.** Services distributed with the platform and basic in configuring and deploying applications. This category would include the user registration service and security services (authentication, authorization, integrity, confidentiality and non-repudiation), where the last of...
these has a transversal affect on all the platform’s components.

- **Specific services.** Services used by applications to configure specific scenarios. One example of this type of services is voting, which is performed through telematic services, while maintaining and guaranteeing compliance with requirements of voting at all times [12].

Services interact with the platform through the second level, which will call the **Operation and Communication Interface.** Given that the possibility is envisaged of these services being provided by external entities, this interface plays an important role in making access to all services uniform.

The third level is the **Operation Manager,** which executes operations in the platform by invoking the aforementioned services. This architecture has taken the form of the design shown in Figure 2.

This model breaks down into the following functional blocks:

- **Access system.** Allows for communication between the user and **platform operations.** Receives requests from users and prepares and calls up the necessary operations of the platform. When the platform responds, it transmits results to the petitioner. Each of these applications is a possible implementation of a specific scenario.

- **Operation Manager.** This is the core of the platform. Its primary mission is to ensure that applications perform their tasks efficiently and that applications can be generated efficiently by developers by providing them with a set of platform operations that is sufficiently abstract and versatile. These operations are broken down into services, and one of the tasks of the Operation Manager, through the **Service Manager,** is to obtain plans for performing these operations and to control their execution through the **Service Invoker.** The latter invokes the proper services at the right time. The **Request Manager,** which sends operations in order to the Service Manager, receives the response and returns it to the application. This task is performed with tools for service orchestration based on stable recommendations. In particular, we have decided to use WSBPEL (OASIS [13]), although implementation will depend on the status of the existing development tools.

- **Application Server.** Although it is not, strictly speaking, part of the platform, this server will act as a container for deploying both web services and the tool packages necessary to perform these services. Section 4 herein defines the specific tools needed to generate these applications and the server used.

- **Security Mechanisms** This functional block will transversally control all the security policies defined in the platform.

### 4. Model of implementation and technology considerations

In this section, we will describe the model of operations used by applications, and discuss certain issues regarding implementation and the technologies being used in the platform.

The W3C consortium provides a series of recommendations to implement the protocol stack of an SOA through **web services** [14]. These recommendations describe the interactions between low-level components, with a high degree of agreement in this area, as this technology is seen as well-established. Substantial work is under way to specify strategies to define interactions...
between high-level components, so as to increase the degree of abstraction in the use of services and transferring abstraction to the application domain.

In defining the conditions and order in which messages are exchanged – that is, the role of coordination - the composition and aggregation of services, aspects which are at the top of the protocols stack of web services, use is generally made of the terms choreography and orchestration.

A choreography is an abstract definition of the collaboration between two entities (peer-to-peer), or: a multi-party contract that describes from global viewpoint the external observable behavior across multiple clients (which are generally Web Services but not exclusively so) in which external observable behavior is defined as the presence or absence of messages that are exchanged between a Web Service and its clients. [15]. Among those engaged in the most work on service choreography are W3C, which has generated three recommendations (now at the working draft stage [16] [15] and Candidate Recommendation [17]) as part of the W3C Web Services Choreography Working Group, whose mission is to define a language called WS-CDL, which is based on WSDL 2.0, to describe a global peer-to-peer model for interaction between companies.

Unlike choreography, where the center of attention is the abstract global model, orchestration focuses on executable business processes. That is, an orchestration models the business process of each of the entities participating in a choreography. Practically, this takes the form of a new web service based on the composition of other web services.

The standard language for orchestrating services is BPEL4WS (Business Process Language for Web Services) (version 1.1, stable) [18] or, the more recent WS-BPEL (version 2.0, presently in draft version). Hereinafter, we will use BPEL as a term to refer interchangeably to one or another version. BPEL was developed by IBM, Microsoft and BEA Systems, by unifying WSFL and XLANG. It is now being standardized by OASIS and there are development tools for version 1.1.

BPEL is an extensible language based on workflow strategies that enables compositions of services with WSDL definitions [19] of them. In essence, a BPEL process uses one or more services described in WSDL to achieve more complex global performance based on messages exchanged with individual services through web service interfaces.

Like WSDL, the BPEL model maintains a separation between the abstract content of the message and the deployment information. That is, it keeps information on partners and their interactions in abstract terms, in a way similar to WSDL (messages, portTypes, operations), without specific references to present services used by the instances of the process. This separation allows a BPEL process to become a reusable definition that can be deployed in different forms and in different scenarios by maintaining behavior at application level (abstract component).

Thus, in each business process modeled with BPEL, there will be two files: a WSDL description of the service composed, with prior information and its relationship to the services it uses; and a BPEL description, which, in addition to bearing information for identifying and locating elements of the services used, contains a set of tags to enable sequencing and ordering message exchange, establishing the workflow of the business process between two partners. Activities in a BPEL process can be broken down into two groups:

- Basic activities. These refer mainly to processing orders such as: <invoke>, <wait>, <reply>, <receive>, <assign> and <empty>.
- Structural activities. These regulate workflow of operations. These resemble control structures used in programming languages. The most important ones are: <sequence>, <switch>, <while>, <pick> and <flow>.

Performance of platform operations has been developed with BPEL. Figure 3 shows the processing of a request from a system user, its relationship with elements of the platform and its process of transformation until a result is obtained.
A user request goes through the following steps:

1. Users call up application services through what we call application service requests. This involves the use of J2EE-related technologies, that is, a set of web pages in a format compatible with the user interface and JSPs or servlets that respond to the user request, dialogue with the user to obtain related information, maintain the session, etc., and then generate the operation, launch it on the platform and wait for the reply.

2. Each of the platform operations is configured as an XML document, with the following sections:
   - Operation name.
   - Parameters.
   - Petitioner ID and role.

3. The platform receives the operations and through a servlet (the Parser in Figure 3) queries the database for previously established plans for each platform operation. These plans are constituted by two descriptions, in accordance with the recommendation for orchestration of BPEL services: a document in WSDL of the new service and the BPEL descriptor of the service. The parser will combine the operations and generate these descriptions which, in this first version of the prototype, will be developed and deployed in the platform as services.

4. The Parser will invoke the BPEL process with the engine. This engine will be in a Tomcat container and enable managing of clients with an Apache SOAP and deployment of web services.

5. As a result of the execution of the BPEL process, operations of platform services will be performed to carry out the operation, and results will be generated for processing by the Results Generator (part of the Request Manager, see Figure 2) and sent to the user who generated the request.

6. Platform services will be deployed and their WSDL descriptions will be accessible to the parser and the engine.

Figure 4 provides a graphic summary of how information is transformed by the system.

Technologies for developing web services are sufficiently well established and many products are available, both proprietary and open source, that enable development of these applications in all phases, to a greater or lesser extent.

In development environment for orchestration of services, it would be interesting to have two tools:

- An editor to facilitate generation and edition of BPEL specifications.
- An engine for execution of BPEL services included as a container, which should be open source if possible, like Tomcat.

To develop the platform, we have installed tools by ActiveBpel, an organization that distributes ActiveBPEL™ [20], an open source execution environment capable of executing BPEL4WS 1.1 processes in real time, and which also has a design environment that can be integrated in Eclipse.

5. Description of test case

In this phase of development, the specific services of the platform are defined with the objective of testing the viability of the platform itself. In an initial, informal use of these services, we will describe their behavior and identify the requirements it must meet and the participants’ roles, and then continue by describing the operations and their workflows.

Although the platform has been conceived to accept the addition of new applications as required in scenarios of use, we believe that applications for forums, blogs, notice boards, voting, wikis and general handling of the census will play a crucial role and will be required in a majority of cases.
To provide these services to the community using the platform, there must be censuses of authorized persons. These censuses must have information for a particular community: for example, in a municipality, a person’s address is important information, while in an academic environment, greater importance may be assigned to the courses attended or taught; in industry, perhaps the department in which a person works, etc. In any of these cases, the census must be the place to store the certificate or public key to provide security guarantees that differentiate between the services of the platform from the use made of them in broadly uncontrolled environments obtaining at present.

The roles for interaction with the platform are the following:

- **Participant or user**: member of the community authorized to make use of the applications offered, that is, participate in the forum, keep a blog, read news, write in wikis, etc.

- **Authorities**: members responsible for supervising use of resources to enforce compliance with established rules.

It has been envisaged that, under certain circumstances, users can participate anonymously - to express an opinion in a forum, blog or write in a wiki - while providing other participants a guarantee that behind an alias is a valid user member of the community. The relationship between the alias and the member of the community is established through cryptographic methods which guarantee that the application will not contain information allowing for this relationship to be established, unless the user wishes to disclose their true identity.

Owing to its importance for all services, the example discussed here is the census web service (WS_CEN), for which seven service operations have been identified enabling location of the census, querying it, adding, deleting or modifying data and adding or removing users.

<table>
<thead>
<tr>
<th>Table 1. Operations of census web service</th>
</tr>
</thead>
<tbody>
<tr>
<td>CEN_Find_Census</td>
</tr>
<tr>
<td>CEN_Add_Data</td>
</tr>
<tr>
<td>CEN_Update_Data</td>
</tr>
</tbody>
</table>

An example of an application-level operation that uses the census web service would be the **PO_Fulfill_Census**, which enables adding, deleting or modifying user census data in an application. For all application-level operations supported by the platform, we have made an informal specification, including the invocation level, description, behavior, the parameters needed, the web services involved in its operation and the result that should be produced by its execution. Figure 5 illustrates this example.

On the basis of an informal operation specification, a pseudo-code is generated defining behavior, so then a workflow can be generated of it in terms of BPEL process elements, as observed in Figure 6 for the operation PO_Fulfill_Census.

```
PO_FULFILL_CENSUS
Invocation level: Boss
Description: Add, delete or modify. Data on participants in applications and in any of the local censuses handled by the application. It must handle the following data set:

PARAMETERS:
Application: Unique ID of a debate application in the platform.
Operation type: Indicates the type of operation to be made on the census:
1 = add additional data to a user
2 = delete additional data from a user
3 = modify census data in a user
Census identification: Unique code identifying census in platform.
User ID: Unique ID of user in census.
User (s) data: Set of census information of a user or subset that can be modified.
User additional data: Set of non-census information or subset that can be added, modified or deleted. For example, e-mail addresses.

RESP_USER_MANAGEMENT_CENSUS
Operation result: A modified local census if the operation has been performed, or an error otherwise.

Behavior
- The system must verify that the census selected is assigned to the application.
- The user ID will be the discriminating factor to locate users in the census.
- ...
```

Figure 5. Informal specification of an operation

The next and final step in executing the operation in the platform is to make a description of the process in terms of WSDL files and the BPEL process, and then the
services are deployed in the container selected for implementation.

This research group has therefore used the emerging technologies of choreography and orchestration of services when designing the platform, defining collaboration between entities in an abstract way. This facilitates the creation of new Web services through the composition of other services in existence, thus achieving a more complex global performance. The independence BPEL maintains between partners and their interactions, in abstract terms, enables a BPEL process to become a reusable definition which can be deployed in different forms and in different scenarios while maintaining application level, or abstract component performance.

In the immediate future, the work of this group will focus on completing implementation of the platform and on conducting a complete case study with BPEL techniques. Nevertheless, the work developed so far shows that service orchestration technologies to implement business processes in Web services are sufficiently mature to consider their use to be natural in the development of applications.

Although it is true that the status of standardization continues to be somewhat confusing, owing mainly to the changes in versions of WDSL and WS-BPEL and the stagnation of recommendations from W3C, it is to be hoped that the situation will be clarified this year, and tools and environments should adapt to the new versions in the near future. For the moment, the tools available are sufficiently powerful to be adapted to present needs.

However work in these technologies is far from complete. There remains broad room for research, particularly in the field of relationships between orchestrations, choreographies and web semantics. Standards organizations, companies and open source organizations are working in this direction.

7. References
EATIS 2007


