

# A simple model and a distributed architecture for realizing one-stop e-government

Gonçalo Paiva Dias <sup>a,\*</sup>, José Aberto Rafael <sup>b</sup>

<sup>a</sup> *Escola Superior de Tecnologia e Gestão de Águeda, Universidade de Aveiro, Zona Industrial da Alagoa, Apartado 473, 3754-909 Águeda, Portugal*

<sup>b</sup> *Departamento de Electrónica e Telecomunicações, Universidade de Aveiro, Portugal*

Received 7 December 2004; received in revised form 20 September 2005; accepted 7 February 2006

Available online 24 March 2006

## Abstract

In this paper, we propose a generic model for one-stop e-government and a distributed architecture for its implementation. The model follows a very basic paradigm: the public administration is composed of an unstructured network of entities that upload and download information objects to/from each other and to/from service repositories in order to deliver client-centered services. The architecture is based on widely available technologies such as HTTP, SSL, XML and PKI and it supports the implementation of life events, single access points, concurrent providers and integrated delivery channels. We argue that the proposed architecture permits the interconnection of almost any kind of government body and that it establishes a common ground upon which new standardization levels can be built. As a starting point, we also define a set of general requirements for one-stop e-government.

© 2006 Elsevier B.V. All rights reserved.

*Keywords:* E-government; One-stop government; Electronic services; Web services; E-government models and architectures

## 1. Introduction

The concepts presented in this paper are the results of our research in the area of e-government; more precisely, they are the outcome of a conceptualization task drawing on our experience of developing prototypes for electronic service delivery at the local authorities' level. Despite the initial restricted scope of our research, the search for general solutions led us to the definition of a generic model for one-stop e-government. In this paper, we describe this model and the distributed architecture that implements it.

We define one-stop e-government as the use of information and communication technologies (ICT) to support the provision of client-centered public services. This definition is based on both definitions of e-government and one-stop government. The European Commission defines e-government as “the use of ICT in public administrations combined with organizational change and new skills in order

to improve public services and democratic processes and strengthen support to public policies” [1]. One-stop government refers to the integration of public services from a customer's point of view [2].

Various architectures for e-government have been proposed as part of both completed and ongoing research projects. Examples include the OSCI platform [3], the SeCo container [4], the AIPA reference architecture [5], the eGOV project [6], the eMayor project [7] and the WebDG architecture [8]. However, although pertinent, the architectures proposed by those projects do not fully address all the relevant requirements of one-stop e-government. When seen individually, they fail to address at least some of the following requirements: the full support of a client-centered approach; the integration of both traditional and on-line delivery channels; the support for concurrent access points; versatility; and security. The model presented in this article aims to address all the listed requirements. It is, nevertheless, based on some of the results achieved by the previously referenced projects, mainly the SeCo and the eGOV projects.

\* Corresponding author. Tel.: +351 234 603839; fax: +351 234 603838.  
E-mail address: [gpd@estga.ua.pt](mailto:gpd@estga.ua.pt) (G.P. Dias).

The rest of this paper is organized as follows. A list of general requirements for one-stop e-government is presented in Section 2. The generic model is presented in Section 3 and the distributed architecture is described in Section 4. In Section 5 both the model and the architecture are discussed by revisiting the requirements previously defined. Finally, in Section 6, the general conclusions are drawn, future research topics are identified, and some final considerations are made.

## 2. Requirements for a generic model

### 2.1. Client-centered approach

The need for a client-centered approach in government was originally asserted in the context of the New Public Management movements [9] and has had a decisive influence in the e-government arena. In effect, many authors have stressed the usefulness of e-government for integrating the vertical silos of government (see for example [10–12]). One-stop government accomplishes client centering by integrating services from the client perspective, either by coordinating processes at the front office or by integrating them at the back-office [2,13].

The essence of the problem is that governments tend to organize themselves in vertical departments that follow their administrative logic and not the users' needs. Since these different vertical departments do not easily communicate with each other, it frequently happens that the users have to contact different departments in order to solve any given problem that requires government intervention. Moreover, government structure is usually too complex to be easily perceived by a common user who is probably much more interested in solving his problems in a quick and efficient way than to understand the administration's peculiarities. As a consequence, users do not always know the exact procedure for resolving any given problem. This functional opacity combined with the need to contact several departments usually results in excessive time and effort being spent by the common citizen or company.

The different government departments should develop the capacity to communicate with each other in such a way that users could benefit from horizontally integrated services. Ideally, a user should be able to make a single contact with a single government access point, whether real or virtual, and fully resolve the query. Consequently, any generic model for one-stop e-government has to take into account that final services should be integrated in a form suitable for the users' needs, following paradigms like life events [13] or others, and that single access points to the government should be supported.

### 2.2. Channel multiplicity

Face-to-face service provision is many times inconvenient for the user and always expensive for the provider. However, as we shall explain, this does not mean that government can

fully substitute its physical presence or other traditional channels by on-line services based on new technologies.

Government services must be accessible to all citizens, even if those citizens do not know how to use a computer, a kiosk or a cellular phone; are unable to read or write or even to understand the available languages; or have any other handicap that prevents them from using one or more delivery channels. It also happens that not all government services can simply be delivered on-line. It is not conceivable that all medical surgeries or police operations, for example, can be fully performed on-line, at least in the near future. Consequently, channels based on new technologies will probably develop and become common, but at least a residual part of the government services will always be delivered by traditional means.

Whenever possible, every government service should be offered through several different channels, including traditional channels like face-to-face, mail, phone or fax, and new electronic channels like the web, electronic mail, video-conference and others. A conjugation of different delivery channels should also be possible if several contacts between the user and the provider are necessary within the provision of a single service. It should be up to the user to choose which one to use in each occasion.

Based upon this view, any model for the electronic delivery of public services has to accommodate the fact that a multiplicity of channels will exist, including both electronic and non-electronic channels. Measures have to be taken to prevent that this multiplicity of channels would result in any unnecessary duplication of means and effort, both in the front-office and the back-office.

### 2.3. Concurrent access points

Different levels of government with different degrees of autonomy are present in almost every country. The central government, the state or regional government and the local authorities are examples of such levels. Depending on the country, some or all of these levels of government have elected officials and, as a consequence, can have autonomous political agendas. Because nation states need to cooperate with each other in order to have effective policies in a globalized world, supra-national levels of government are also under development. The European Union is probably the best example of this trend, known as the network state [14] or multi-level governance [15].

Besides existing at different levels, governments also cooperate with non-governmental institutions both in the formulation of their policies and in the provision of their services. This is usually referred to as public governance [16]. In the last two decades this has been reinforced by the privatization and sub-contracting of some of the traditional functions of government, largely influenced by the New Public Management movements [9].

In order to fulfill the user's needs, cooperation between the different levels of government and the subsidiary non-public institutions is mandatory. Obviously, the need for this

cooperation has to be taken into account when developing models of e-government. It is also important to note that, in this context, several access points to the administration will probably have to coexist. Examples include the provision of local or regional services through a national or supra-national web based portal; the provision of national or supra-national services through regional or local access points; the provision of government services by private partners, eventually with added value; and the provision of private or semi-private services through government access points. It is also important to note that elected bodies will probably never renounce the capacity to directly provide services to their own citizens, although they will probably admit that complementary service channels exist.

#### 2.4. Versatility

In a broad sense, the provision of government services includes several levels of government with their different departments and their non-public partners. This creates a heterogenic network of entities, each with its own culture, management practices and degree of technological maturity. If technology is to be used to successfully integrate services in this network, special care has to be taken to understand their differences, to respect their timings and not to disrupt their work. Instant, simultaneous and full normalization is, consequently, out of the question.

Although full standardization will not be achieved in a single instant in time, some degree of normalization will always be required in order to integrate anything at all. The least or the biggest common denominators approaches could be applied but they would, respectively, prevent innovation or become too complex to be of any use; the point of equilibrium has to be achieved somewhere in the middle. Moreover, this point of equilibrium will tend to change over time, as technology adoption grows and new technologies become available. To be successful, models should reflect these considerations.

#### 2.5. Security

Security is a critical success condition and, simultaneously, one of the main challenges of e-government. Several factors contribute to this: the absolute need to assure user confidence; the large spectrum of potential users; the need to conciliate legal constraints relating to data privacy and freedom of information; the diversity of government processes with different security constraints; the diversity of entities that provide those services; and the typical usage of distributed environments and public communication infrastructures to implement e-government.

Several alternative methods have been proposed for the gathering of security requirements in e-government (see for example [17,18]). Nevertheless, although security requirements may vary with the concrete application, the majority of authors agree on the relevance of at least some of the following general requirements [18–22]:

- Authentication, as the ability to properly and securely identify system users.
- Authorization, as the ability to grant access privileges to the resources based on the available user's identities.
- Confidentiality, as the capacity to prevent information from being accessed or understood by unauthorized users or systems.
- Integrity, as the ability to prevent information from being intentionally or unintentionally corrupted or tampered with.
- Availability, as the ability to grant access to resources within a reasonable period of time.
- Non-repudiation, as the ability to prevent any user from later denying his intervention in a given process or transaction.

In order to be useful in practice, models for one-stop e-government have to address at least these general requirements. More specialized requirements that depend on the concrete application and that, thus, cannot be easily anticipated will have to be implemented combining the basic mechanisms of the platform or addressed using alternative approaches.

### 3. The model

#### 3.1. Entities

The model follows a very basic paradigm: the public administration is composed of an unstructured network of entities that exchange electronic requests in order to deliver services. An entity is an abstract concept that includes every institution or department that provides services to the users (citizens, businesses or public officials as clients of public services), whether directly or indirectly. If a user asks for a given service to a given entity, this entity will either be capable of providing the service itself or it will try to find an entity that can do so. If such an entity is available, it will request the service to this other entity instead of asking the user to do so. Moreover, instead of asking the users for certificates or other official documents, entities will request each other for those documents whenever they need them. Complementarily, entities will also be capable of querying each other for request states and results.

Users request services, query their state, and receive results through front-office entities. Examples of such entities include a citizen's portal, a national one-stop shop or a local authority front office. Nevertheless, the concrete nature of the entities is purposely left abstract at the model level. They can be materialized, for example, by a fully automated portal based on agent technology that is able to receive and automatically route user requests, by a manually configured portal that only offers a set of pre-defined services, or by a physical balcony in which humans act as intermediaries between the users and the system.

Front-office entities rely on back-office entities to deliver services. Back-office entities accept and process requests

and retrieve their results to front-office entities so that they can be forwarded to the users. The abstract nature of entities also allows very different degrees of technological maturity at the back-office. Examples include a fully automated central authority; a semi-automated specialized regional office or a mainly human based local authority back-office. An entity can be simultaneously a front-office entity, that provides services directly to the public, and a back-office entity, that delivers services through other entities.

### 3.2. Service repositories

Every entity must know which services it is capable of providing and must be able to publish metadata that describe these services in one or more service repositories. Thus, service repositories are collections of service descriptions that include all the relevant information needed for entities to request services from other entities. Requesting entities will query these service repositories in order to find the available services. This concept of service repository is based on the one originally proposed by the eGOV project [6].

It is not a concern of the model how, in practice, requesting entities will benefit from service repositories. They can be used to dynamically find and request services, whether automatically or by human intervention; or they can be used as a means of finding information about services that will be statically configured as available at the requesting entities side. Service repositories can be installed in the participating entities or in autonomous nodes. Several service repositories can co-exist in the same network and no centralized unit is mandatory.

### 3.3. Services, requests and reports

Entities communicate with each other and with the repositories by uploading and downloading services, requests and reports. Services are descriptions of what can be requested and how. They are uploaded to the repositories by the providing entities and are downloaded from there by requesting entities.

Requests are instances of services that requesting entities create and upload to providing entities when they need to request a service. The providing entities keep a record of the successive events and results of the requests they receive and, whenever needed, they create reports to describe their state. Requesting entities can download reports at any time to check the state and the available results of the requests. Optionally, providing entities may upload reports to the original requesting entities as a means of notifying them of an event occurrence.

The only assumptions made relating to the internal organization of the entities are that they must know the services they provide, adhere to a standardized and secure communication platform and be able to upload and download the defined objects using normalized formats.

### 3.4. Security

Services, requests and reports are digitally signed in order to authenticate their issuers and prevent their tampering or repudiation. User authentication is a responsibility of the entities that issue the requests. Whenever applicable, requests will include user authentication and authorization related information in order to allow the entities that received them to grant or deny access to the relevant resources. A trust chain will be established between participating entities by means of a public key infrastructure (PKI). Services, requests and reports are uploaded and downloaded to/from entities using privacy-preserving protocols. Availability is an implementation specific concern that, thus, has to be addressed at that particular level.

### 3.5. Document formats

Since requests are the objects that trigger the provision of services, they must include all the relevant information for that purpose. This can include forms and many other types of documents. Thus, to be universal, requests must be capable of incorporating virtually any type of digital document. The same could be argued with respect to results within report objects or to form definitions within service objects. Consequently, we assume that any implementation of these objects will support the incorporation of unstructured binary data.

Although the model does not intrinsically depend upon the formats used to transfer digital documents, the availability of standardized formats cannot be ignored. In an ultimate analysis, if the entities were not capable of interpreting the transferred documents, they would be unable to provide the actual services. Consequently, the existence of standardized document formats constitutes another of our assumptions. Fortunately, many standards are available in the market for the most varied applications, forms included. Which format should be used in each particular situation is a service specific detail that should thus be included as metadata in the service object itself.

## 4. The distributed architecture

Fig. 1 depicts the general architecture that supports the defined model. Three types of nodes are available in the network: requesters, providers and service repositories. Requesters are associated with entities that request services (see entities A and B) and providers are associated with entities that provide services (see entity C in the figure). Service repositories can be associated with a requester, with a provider or deployed in a standalone mode (as it is represented in the figure). If an entity is simultaneously a service requester and a service provider, it will be associated with both a requester and a provider. Requesters and providers communicate with each other and with the service repositories by uploading and downloading services, requests and reports.

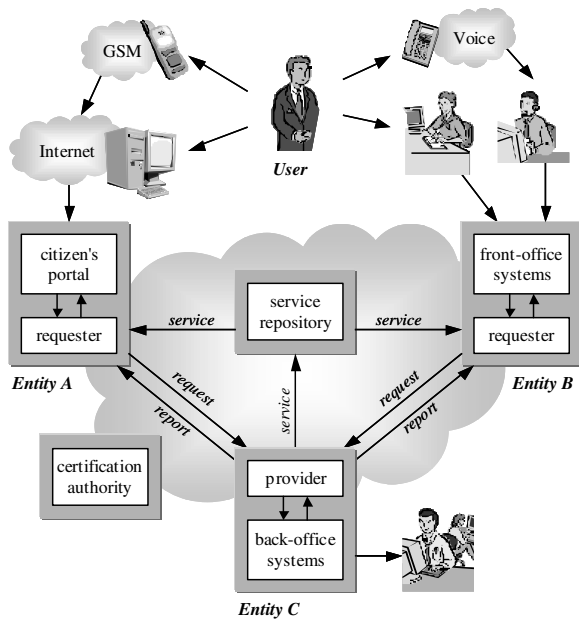


Fig. 1. The general architecture.

Entity A in the figure represents a portal implementation while entity B represents a one-stop-shop implementation. Both entities use identical requesters to access the service repository and the services provided by entity C. As represented, the conjugation of A and B allows the user to request or query the state of the services provided by C using a set of alternative channels: mobile phone (WAP/UMTS), personal computer, phone, and face-to-face. The same result could be accomplished by having a single requesting entity that would concentrate all the represented channels. Other channels could additionally be represented.

Although not represented in the figure, several service repositories and several providing entities can co-exist in the same network. If this is the case, the requesting entities can integrate services published in several repositories and/or provided by different entities. It is also possible to have entities that provide services both to the public and to other entities, and entities that simultaneously provide and request services. A certification authority responsible for issuing the digital certificates that authenticate the participating entities is also represented in the figure.

#### 4.1. Requesters and providers

The requester is a black box that isolates the local systems from the complexity of the network and the network from the specificities of the local systems. It implements a normalized interface with methods that can be invoked by the local systems to get services, send requests, and get reports (downward arrow from local systems to requester in entities A and B). It stores reports that are received from the network by using a symmetric mechanism: it invokes methods from one or more interfaces implemented

by the local systems (upward arrow from requester to local systems). These interfaces have to be developed using a well-defined set of rules so that the requester can be configured to use them correctly. Standardized interfaces can be built for the more common platforms. Configurable interfaces can be used if a messaging environment is available.

The provider works for providing entities much in the same way that the requester does for requesting entities. It uses locally implemented interfaces to store information about incoming requests and to collect information about previously stored requests (downward arrow in entity C), and it implements a normalized interface that allows the local systems to publish services and send reports (upward arrow).

#### 4.2. Service repositories

A service repository is a warehouse where providers publish service metadata so that it can be searched by requesters. Where a repository is localized and what kind of services it should store depend on the concrete application. In the case of a citizen portal, for example, it may be suitable to have a repository associated with the portal requester, so that every provider that contributes with services can publish them there. For a central department that provides services to many requesters, it may be practical to have the repository associated with the providing entity. Thematic service repositories used by several requesters and providers may also be useful: services to local authorities, health related services, building related services, etc. Many service repositories can co-exist. The same service can be published in more than one repository and requesters can aggregate information from queries to several repositories. Requesters are locally configured with the list of repositories they are allowed to query and providing entities explicitly define in which repositories each service should be published.

#### 4.3. Transferable objects

Requesters and providers communicate with each other and with the service repositories by uploading and downloading three different types of standardized objects: services, requests and reports. The class diagram in Fig. 2 shows the existing associations between these objects: each service can be associated with multiple requests, while each request must be associated with a single service; each request can be associated with multiple reports, while each report must be associated with a single request. Providers are responsible for generating the unique identifications of the services they publish, the requests they receive and the reports they produce. Services have universal identifications. Requests are fully identifiable within the entities that provide them. Reports depend on their parent request to be fully identifiable.

Service objects consist of metadata. They include descriptions of the real services that they represent (their aim, which entities provide them, who can request them,

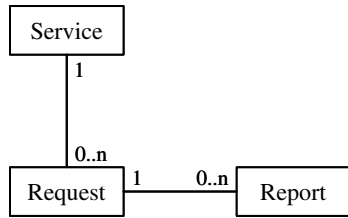


Fig. 2. Transferable objects associations.

what results may be expected, etc), and structured information about how to actually request the service (provider’s digital address, mandatory forms, attachable documents, recognized formats, available payment options, etc).

Requests are objects that aggregate all the information relevant to start the provision of a specific service. They include three different sections of data: a details section, whose structure is independent of the actual service being requested; a form-list section, that includes one or more mandatory forms; and an attachment-list section, that includes the documents attached by the user. The details section includes information about the user (his identity and authentication and authorization related data), the requesting entity, and the specific request being transferred (related service, payment method, payment data, result notification method, etc).

Reports are representations of the state of a given request at a specific instant in time. They include information collected during the process of requesting and providing a concrete service: which user requests them, through which requesting entity, their current and past states, the relevant dates, etc. They may also include a list of intermediary or final results: receipts, notifications, certificates, etc.

4.4. Network interactions

Requesters and providers use the HTTP protocol (hypertext transfer protocol) [23] over the SSL protocol (secure sockets layer) [24] to transfer objects between themselves and the repositories. The POST command is used to upload services, requests and reports, and the GET command is used to find and download services and reports. In order to be sent over the network, the objects are coded into XML (extensible markup language) [25] tags and embedded into XML containers. Thus, what is actually uploaded and downloaded are these XML containers, and not the objects alone.

The upper diagram in Fig. 3 shows a valid interaction between a provider and a repository. Providers publish new services by sending a POST command with a service object. The object is embedded in an XML container (data coded into XML containers is represented in the figure between square brackets). The repository acknowledges the reception of the new service by sending a synchronous confirmation response or, eventually, an error code. Services can be modified by being overwritten by new services and deleted by being overwritten by void services.

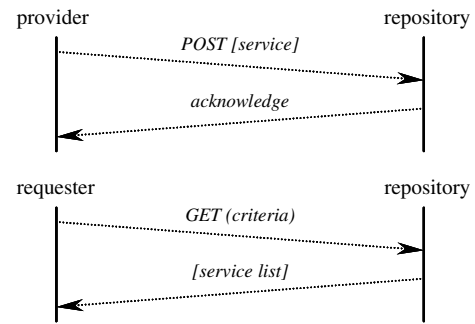


Fig. 3. Interactions with the repositories.

The lower diagram in Fig. 3 shows the interaction between requesters and repositories. Requesters query the repositories by sending a GET command with search criteria. These search criteria are coded into the URI (uniform resource identifier) [26] by using the URI query component (data coded into the URI is represented in the figure between round brackets). The repositories use the search criteria to find the relevant services and, once they are found, a synchronous response with the list of services is sent. This list is also embedded in a XML container. A void list is sent if no services are found and an error code is sent if an exception occurs.

The diagrams in Fig. 4 show the possible interactions between requesters and providers. The first diagram from the top shows how a new request is uploaded from the requester to the provider. The request object is embedded into a XML container and sent using the POST command. The provider acknowledges the reception of the new request by sending back a report with the provider given identification of the request, its current state and a list of any immediately available results.

The second diagram shows the interactions that take place when requesters download reports from providers. A GET command with the search criteria is sent. The provider uses these search criteria to find the relevant requests, collects

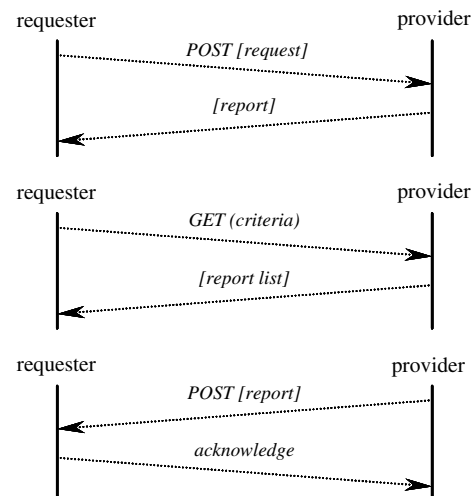


Fig. 4. Interactions between requesters and providers.

information about their state and results, and sends back a list of reports. An empty list or an error code is retrieved if, respectively, no requests are found or an exception occurs.

The last diagram shows the case when, after some state transitions, providers upload reports to the original requesters. This only happens when the request itself indicates that the original requester expects this behavior from the provider. Because this behavior may not always be possible, service repositories include information about the availability of this option for each service.

#### 4.5. The container structure

Requesters and providers upload and download services, requests and reports. In order to be transferred, these objects are embedded in secure XML containers. The basic structure of the XML container is based on that proposed by the Secure Electronic Contracts (SeCo) project [4]. It includes a content block and a signature block. The content block is the recipient where objects are inserted. The signature block holds the digital signature that signs the content block, together with the X.509v3 digital certificate [27] that holds the public key of the signer. This binary information is embedded into the signature block by using BASE64 encoding [28]. Thus, the container itself contains the mechanisms that assure the integrity and the non-repudiation of its content. The privacy of the content is assured by the use of the SSL protocol to transfer the containers. An X.509v3 PKI (public key infrastructure) is used to authenticate the participating entities.

What is actually inside the content block depends on what is being transferred in each occasion. Fig. 5 depicts the detailed container structure when a request is being uploaded to a provider. The content block embeds a request object composed of a details section, a form-list section and an attachment-list section. The form-list section contains the mandatory forms and the attachment-list section contains a list of attached documents. Both the forms and the attachments are encoded with BASE64 so that they can be embedded into the XML container. They can be individually signed by their original authors in a way similar to that of the containers, however, this is a function that must be performed by the entities themselves and not by requesters or providers. Requesters and providers may encode and decode documents and forms but they do not interpret them at all. They are passed “as they are” from the local systems of one entity to the local systems of the other entity. It is up to the entities to assure that these documents will be correctly interpreted. The metadata downloaded from the service repositories and passed to the entities includes information about which formats are admissible for each service.

The structure of the container when a report is being transferred is similar to the one of the request container. The content block includes a details section, a log section and a results list. The details section identifies the related request, the log section describes the current and previous

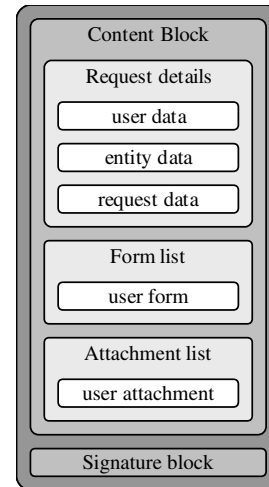


Fig. 5. Request container structure.

states of the request, and the results section includes a list of the currently available results. The signing of the individual results is possible but similar considerations to those of the request container are applied.

## 5. Requirements revisited

The presented architecture and the underlying model offer a solution that can smooth the implementation of one-stop e-government. The paradigm is simple and universal: service integration results from the cooperation of the several entities. Elegant solutions are offered to build single access points and to accommodate the co-existence of concurrent providers in the same network. The integration of different delivery channels is accomplished by interfacing the local systems and, consequently, does not necessarily imply duplication of means. Security mechanisms are built into the architecture and can be implemented using available technologies.

### 5.1. Life events

The implementation of the life event paradigm needs further discussion. We identify two different perspectives relating to life-events: strong and weak implementations. By a strong implementation we mean the provision of integrated services that have a univocal correspondence with one life event. More precisely, this implies that these are services through which the user gets all the necessary government results relating to a life event resulting from a single integrated request. A strong implementation of the moving house life event, for example, would imply that with a single request and, ideally, a single form, the user could inform all relevant departments of his new address, obtain his new personal documents, transfer his water and electricity supply contracts, obtain a residential parking permit, or get any other relevant results.

By weak implementation it is inferred the simple cataloging of fairly independent services using a life event

paradigm. The services are presented logically together but each one is associated with a single request and has independent results. In a weak implementation, the previous example would allow the user to find all relevant information about the services related to moving house, but those services would still have to be requested in a way independent from each other.

The implementation of a life event using the presented architecture depends on the perspective used. Weak life events can be implemented simply by using the service repositories capabilities. Each individual service can be qualified in such a way that life event oriented queries would return all the relevant services for each life event situation.

Strong life events need more capabilities than the ones provided by the service repositories themselves. The need to decompose a complex service, possibly with several options, direct the individual parts to the relevant entities, and receive and integrate the relevant results requires workflow management and, thus, in the scope of the presented architecture, also requires a responsible entity. Two approaches can be applied: the entity responsible for providing the life event oriented services relies solely on the information provided by the service repositories or it possesses additional knowledge about those services. Using the first approach the development of fully automated entities can be explored, however, the provision of the more complex life events will probably benefit from external knowledge and thus, independent entity configuration.

### 5.2. Normalized services

For an efficient performance some applications may need further normalization than so far presented. A change request to the user's address which is directed to several entities at the same time constitutes a good example of this necessity. In effect, the process of requesting the service would become much simpler if the same service description and, consequently, the same form and the same document formats were shared by all the providing entities. Although the presented architecture does not impose service normalization, it does not prevent it from being established either. As such, the use of this architecture allows the adoption of a service standardization process where common services can be requested and reported using common formats and layouts. The existence of service repositories can even facilitate this process by allowing several entities to share the same normalized service descriptions. Another advantage is that, with this architecture, the normalized services can be adopted gradually: neither for all services at the same time nor all entities.

### 5.3. Heterogeneity

The presented architecture makes no further assumptions relating to the internal organization of the entities

other than knowing how to use the requesters or providers interfaces and being able to publish their own interfaces in a suitable form to be used by those providers or requesters. This simplicity allows a great diversity of entity structures. To some extent, entities can have an orientation based upon their own idiosyncrasies and take advantage of their existing systems and organization without compromising the overall functionality. Once interfaces are available, the overall system can incorporate entities as diverse as a fully-automated citizen's portal or a mainly paper based local authority. However, it is not a meaningless fact that both the requester and the provider must be locally deployed and, thus, versions suitable for different computer architectures and/or operating systems may have to be developed. Fortunately, the use of technologies like the JVM (Java virtual Machine) [29] may help to alleviate this burden.

### 5.4. Entity authorization

As presented, requesting entities are responsible for the secure authentication of the users. Whenever an authenticated user takes an action that implies issuing a new request, the requesting entity builds a new request object and sends it to the relevant providing entity. This request object includes information about the user and the authentication process that was followed to securely identify him. Providing entities need this authorization related information to manage access control.

Because of the security mechanisms built into the architecture, providing entities can be certain that the authorization information they receive was originally sent by the identified requesting entity and that it has not been tampered with. However, they have no means to verify the accuracy of the information that was originally sent. Consequently, if they do not trust the requesting entity and if access control is vital for the service in question, they have no other choice than to simply reject the request.

Because no other entity authorization mechanisms are available in the architecture, providers need to be statically configured with the list of requesting entities that are trusted, eventually for each service provided. Global entity classes can be created to facilitate the management of such lists: departments of a specific government branch, local authorities, semi-public authorities, etc. Trust can then be based on global classes instead of the individual entities. Class membership can be incorporated as a standard extension to the X509v3 certificates and, thus, be verifiable by providing entities. Service objects can include information about the classes that are allowed to request them. Entities can belong to more than one class and those multiple memberships can be expressed in the same or in different certificates, eventually issued by different certification authorities. Different classifications can co-exist (international, national, sector-based, etc.). These last two possibilities allow the adoption of a gradual normalization process for entity classification.

### 5.5. Financial transactions

One fundamental question that needs further discussion is how to handle payment using the presented architecture. Because the proposed model defines that users access services through front-office entities, the logical extension is to assume that users pay services directly to the front-office entities and that front-office entities transfer payments to back-office entities whenever they request their services. This approach has three important advantages: life-event payment integration can be supported at the front-office; cash and checks remain acceptable payment methods when physical front-offices are used; and user financial information, if any, is kept within the entity that deals directly with the user. However, the feasibility of this approach depends on the availability of an external funds transfer mechanism.

Funds transfers between the participating entities can be made using the banking sector infrastructure. If that is the case, information about the account to credit and the movement description to use for each service provision is included in the report that confirms the reception of each request. Thus, when providing entities accept a request, they send back all the information that the requesting entity needs to make the payment order. To verify if the payment was correctly made, the providing entity checks the credits and their descriptions in its own account. If necessary, a receipt can be included in the final report.

Although the described payment method seems appropriate for the presented model, two other alternatives are available. The first alternative is the transference of the payment order information from the requesting entity to the user, so that the user can make the payment through his own bank. This payment can be made using a debit card and PIN (personal identification number) through the Web, using an automatic teller machine or directly at the bank. Alternatively, credit card information can be inserted into the details section of the request and thus, be transferred from the requesting to the providing entity. This last option, however, has serious security drawbacks: although privacy is assured, the user relinquishes full control regarding what entities the information is sent or what usage is made of it.

The payment options available for each entity and each service, if any, are described in the service objects and, thus, available in service repositories. The actual payment method to use in each occasion may be selected by the requesting entities by filling in the appropriated fields in the request details section.

### 5.6. Standards evolution

The proposed architecture is based on existing standards like HTTP, SSL, XML, BASE64 or X.509v3. It is an important point to consider to what extent any future evolution of these standards will affect the functionality of the overall system and how easily it can incorporate the benefits from those evolutions. The first consideration to make is that the technologies in question are widely used and, consequently, eligi-

ble for state of the art innovations but, simultaneously, not very susceptible to disruption. If retro-compatibility is assured, no major difficulties will arise from the adoption of improved standards. If not, redesign will probably be contained within requesters and providers software, not affecting the entities themselves. If that is the case, the overall cost of redesign will depend mainly on the modularity of the original requesters and providers and, thus, it becomes essentially a software engineering problem.

## 6. Conclusions

The architecture proposed in this article supports the development of one-stop e-government by establishing the basic conditions that allow entities involved in the delivery of governmental services to cooperate with each other for the benefit of the final users. The architecture supports the implementation of life events, single access points, concurrent providers and integrated delivery channels. It is based on widely spread and consolidated technologies and it incorporates solutions for authentication, authorization, confidentiality, integrity and non-repudiation. It is a fully distributed architecture in the sense that all the knowledge about the network is distribute amongst the participating nodes and no centralized unit is mandatory. Finally, it establishes a relatively light common ground that allows for the interconnection of almost any kind of entity and upon which new standardization levels can be built.

Different approaches could have been taken to implement the generic model that is the basis of this architecture. Above all, Web Services technology [30] seems like a promising alternative. In effect, SOAP (simple object access protocol) over HTTP can be used to transfer objects between requesters, providers and repositories, and the use of UDDI (universal description discovery and integration) can allow repositories to be automatically discovered by both requesters and providers. XML Encryption [31] and XML Signature [32] could also have been used for implementing containers. This would allow entities to sign only the relevant parts of the container and, eventually, to sign different parts of the container with different digital signatures. The use of SAML [33] (security assertion markup language) to exchange user authentication and authorization information could also prove to be a valid alternative and, most importantly, could possibly contribute to solve the questions related to entity authorization. Nevertheless, it seems that the proposed model is generic enough to remain valid despite the different technologies that can be used to implement it.

The presented model follows a process-oriented approach instead of an information-oriented one. Consequently, it does not offer solutions for questions such as document searching and retrieval, data manipulation, ad-hoc querying, or alike. Although dynamic process composition may be supported, it is especially suitable for predefined structured processes. Because trust has to be established between the participating entities, it is also more suitable to be used within stable groups of partners

than in open and global partnerships. Asynchronous connections are not supported by the proposed architecture.

The proposed architecture still has some margin for improvement and refinement; subjects like form standardization, service repositories, querying languages, entity authorization, payment methods and interface specification can still be studied further. Another area for investigation is the impact of such a model both in terms of the efficiency and the efficacy of government. How the entities can be organized in order to take the best of the proposed model also remains an open topic of research.

A final consideration is that although the presented architecture and the underlying model offer a solution that facilitates the communication between different entities, they do not eliminate the need for the individual entities to develop this communication ability. The development of a client-centered approach in government implies profound cultural changes. Technology can help promote these changes but it is neither a way of avoiding them nor a panacea for all the problems. It is our conviction that much work still has to be done in this area before e-government can be fully realizable. Nevertheless, it is also our conviction that the existence of simple and feasible models constitutes a relevant contribution to that effort.

## References

- [1] The role of eGovernment for Europe's future, Communication of the European Commission, 2003. <[http://europa.eu.int/information\\_society/eeurope/2005/doc/all\\_about/egov\\_communication\\_en.pdf](http://europa.eu.int/information_society/eeurope/2005/doc/all_about/egov_communication_en.pdf)> (15/09/2005).
- [2] M. Hangen, H. Kubicek (Eds.), *One-stop Government in Europe: Results of 11 National Surveys*, University of Bremen, 2000.
- [3] F. Steimke, M. Hagen, OSCI: a common communication standard for e-government, in: Proc. EGOV'03, Lecture Notes in Computer Science, vol. 2739, Springer, Berlin, 2003, pp. 250–255.
- [4] M. Greunz, B. Schopp, J. Haes, Integrating e-government infrastructures through secure XML document containers, in: Proc. of the 34th Annual Hawaii International Conference on System Sciences (HICSS-34), vol. 5, IEEE, New York, 2001, p. 5004.
- [5] F. Arcieri et al., Experiences and issues in the realization of e-government services, in: Proc. of the 12th International Workshop on Research Issues in Data Engineering: Engineering E-Commerce/E-Business Systems (RIDE'02), IEEE, New York, 2002, pp. 143–148.
- [6] M. Wimmer, A European perspective towards online one-stop government: the eGOV project, *Electronic Commerce Research and Applications* 1 (1) (2002) 92–103.
- [7] A. Kaliontzoglou et al., A secure e-government platform architecture for small to medium sized public organizations, *Electronic Commerce Research and Applications* 4 (2) (2005) 174–186.
- [8] B. Medjahed et al., Infrastructure for e-government web services, *IEEE Internet Computing* 7 (1) (2003) 58–65.
- [9] C. Hood, A public management for all seasons? *Public Administration* 69 (1) (1991) 3–19.
- [10] At the dawn of e-government: the citizen as a customer. Deloitte Research, 2000. <[http://www.deloitte.com/dtt/cda/doc/content/at\\_the\\_dawn\\_of\\_egovernment\(1\)\(1\).pdf](http://www.deloitte.com/dtt/cda/doc/content/at_the_dawn_of_egovernment(1)(1).pdf)> (15/09/2005).
- [11] D. Holmes, *E-gov: E-business Strategies for Government*, Nicholas Brealey Publishing, 2001.
- [12] R. Silcock, What is e-government, *Parliamentary Affairs* 54 (1) (2001) 88–101.
- [13] M. Vintar et al., Delivering better quality public services through life-event portals, in: Proc. of the Tenth NISPAcee Annual Conference: Delivering Public Services in CEE Countries: Trends and Developments. <<http://unpan1.un.org/intradoc/groups/public/documents/nispacee/unpan004382.pdf>> (15/09/2005).
- [14] M. Castells, *End of millennium*, in: *The Information Age: Economy, Society and Culture*, vol. 3, Blackwell, Oxford, 1998.
- [15] A. Smith, Multi-level governance: what it is and how it can be studied, in: G. Peters, J. Pierre (Eds.), *Handbook of Public Administration*, Sage, Beverly Hills, CA, 2003.
- [16] E. Löffler, Governance and government: networking with external stakeholders, in: T. Bovair, E. Löffler (Eds.), *Public Management and Governance*, Routledge, London, 2003.
- [17] M. Wimmer, B. von Bredow, A holistic approach to security aspects in e-government, in: Proc. of the 35th Hawaii International Conference on System Sciences (HICSS-35), vol. 5, IEEE, New York, 2002, p. 128b.
- [18] C. Lambrinouidakis et al., Security requirements for e-government services: a methodological approach for developing a common PKI-based security policy, *Computer Communications* 26 (16) (2003) 1873–1883.
- [19] F. Arcieri et al., A layered IT infrastructure for secure interoperability in Personal Data Registry digital government services, in: Proc. of the 14th International Workshop on Research Issues on Data Engineering: Web Services for E-Commerce and E-Government Applications (RIDE'04), IEEE, New York, 2004, pp. 95–102.
- [20] N. Boudriga, Technical issues in securing e-government, in: Proc. of the 2002 IEEE International Conference on Systems, Man and Cybernetics, vol. 2, IEEE, New York, 2002, pp. 392–397.
- [21] M. Caloyannides et al., US e-government authentication framework and programs, *IT Professional* 5 (3) (2003) 16–21.
- [22] J. Joshi et al., Digital government security infrastructure design challenges, *Computer* 34 (2) (2001) 66–72.
- [23] R. Fielding et al., Hypertext Transfer Protocol – HTTP/1.1, RFC 2616, The Internet Society, 1999. <http://ietfreport.isoc.org/rfc/rfc2616.txt> (15/09/2005).
- [24] A. Frier, P. Karlton, P. Kocher, *The SSL 3.0 Protocol*, Netscape Communications Corp., 1996.
- [25] T. Bray et al. (Eds.), *Extensible Markup Language (XML) 1.0*, third ed., Recommendation, W3C, 2004. <<http://www.w3.org/TR/REC-xml/>> (15/09/2005).
- [26] T. Berners-Lee, R. Fielding, L. Masinter, Uniform Resource Identifiers (URI): Generic Syntax, RFC 2396, The Internet Society, 1998. <http://ietfreport.isoc.org/rfc/rfc2396.txt> (15/09/2005).
- [27] R. Housley, W. Polk, W. Ford, D. Solo, Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, RFC 3280, The Internet Society, 2002. <http://ietfreport.isoc.org/rfc/rfc3280.txt> (15/09/2005).
- [28] S. Josefsson (Ed.), *The Base16, Base32, and Base64 Data Encodings*, RFC 3548, The Internet Society, 2003. <http://ietfreport.isoc.org/rfc/rfc3548.txt> (15/09/2005).
- [29] T. Lindholm, F. Yellin, *The Java Virtual Machine Specification*, second ed., Sun Microsystems, 1999.
- [30] E. Newcomer, *Understanding Web services: XML, WSDL, SOAP, and UDDI*, Addison-Wesley, Reading, MA, 2002.
- [31] D. Eastlake, J. Reagle (Eds.), *XML Encryption Syntax and Processing*, Recommendation, W3C, 2002. <<http://www.w3.org/TR/xmlenc-core/>> (15/09/2005).
- [32] D. Eastlake, J. Reagle, D. Solo (Eds.), *XML Signature Syntax and Processing*, Recommendation, W3C, 2002. <<http://www.w3.org/TR/xmldsig-core/>> (15/09/2005).
- [33] S. Cantor, J. Kemp, E. Maler (Eds.), *Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V 2.0*, Standard, OASIS, 2002.