A Knowledge-Based Approach for Developing Multi-Channel e-Government Services

C. Vassilakis[1], G. Lepouras[1], C. Halatsis[2]
[1] Department of Computer Science and Technology, University of Peloponnese, Terma Karaiskaki 22100, Greece
[2] Department of Informatics, University of Athens, 15784, Greece
{costas, gl}@uop.gr, halatsis@di.uoa.gr

Abstract

Having realised the benefits resulting from delivering on-line public services in the context of electronic government, administrations strive to extend the spectrum of services offered to citizens and enterprises, as well as to engage multiple communication channels in service delivery, in order to increase the target audience and, consequently, the service effectiveness. Insofar, however, only the web channel has been sufficiently used for service delivery, whereas other channels have not been adequately exploited. One of the main reasons of this lag is the cost incurred for the development and maintenance of multiple versions of an electronic service, each version targeted to a different platform. In this paper, we present an approach and the associated tools for developing and maintaining electronic services that allows the automated production of different versions of the electronic service, each targeted to a specific platform.

Keywords: e-government; electronic services; multi-channel service delivery; forms design for eServices

1. Introduction

According to the European Commission, electronic government can be defined as an ever-increasing and pervasive use of information and communication technologies in the context of the Information Society, which more and more affects the public sector; the importance of this development is increasingly acknowledged in many countries around the world and experiments are being conducted at all levels of government -local, regional, national and European- to improve the functioning of public services concerned and to extend their interaction with the outside world [1]. This interaction is mainly performed in order to achieve three goals: providing public services, improving managerial effectiveness, and promoting democracy [2]. For public services provision in particular, benchmarks have been defined [3] and studies have been conducted regarding the development and sophistication of on-line services ([4], [5], [6]).

Besides however promoting on-line availability and sophistication of their services, administrations should also take into account the *channels* through which their services are delivered. Traditionally, administrations deliver their services through the web, while other potential service delivery channels (e.g. WAP, i-mode, SMS, phone centres) are not adequately considered in most cases. This policy though introduces additional barriers to the service acceptance and use by citizens for a number of reasons including:

1. *internet usage*. In order to use some service deployed through the WWW a citizen should have access to the Internet. Since not all potential service users

have access to the internet (studies report that internet usage -either from home or from work- ranges from 66.5 % in North America to 31.6 % in Europe (in average) [7]), limiting a service to the WWW channel effectively excludes a large portion of the population. Although several measures can be taken to alleviate this problem, including information kiosks [8], [9] and access through library computers, the issue cannot be fully addressed, since it does not only relate to computer availability but computer literacy and other factors as well.

2. *channel penetration and user readiness.* Some households and/or enterprises do not own a computer with an internet connection, but most of them are equipped with faxes, phones or mobile phones, with some of them encompassing WAP or i-mode capabilities (e.g. [10], [11]). This portion of potential service users could more readily access a service if it were delivered through one of the latter channels.
3. *lack of appropriate know-how.* Computers are more complex to learn and use than mobile phones, faxes or interactive TV; thus delivering services through the "simplest channels" broadens the target population. An example of such a case is the issuance of tax clearance certificates service, provided by the Greek Ministry of Finance. The service is available via the web and the fax channel, but its target group (which has been limited to certified governmental agencies, notaries and banks to guarantee the confidentiality of taxation data) shows a strong preference towards the fax channel ([12], [13]).
4. *channel appropriateness for specific services*. Some services are inherently more appropriate for delivery through mobile channels (e.g. paying tolls, reserving parking places etc) since citizens mainly use these services on the move, and consequently have limited or no Internet access. On the other hand, a number of complex services may not be appropriate for all platforms, e.g. filling in a tax return form using SMS technology is clearly impractical and should not be considered. For determining whether a service should be delivered through multiple channels (and which these channels are), a six-step methodology is presented in [14]. According to this methodology, firstly the features of the available channels are rated; this is a service-independent step and can be performed only once for all services. Subsequently, service provision requirements for each service are rated and channel features are matched against service provision requirements. Channels meeting all the requirements for a specific service are short-listed. The short list is then filtered, ruling out channels (a) not commonly used by the service target group (b) technically or organisationally unsuitable for the service and (c) not being cost-effective. In the rest of this paper we will only consider services which will be delivered through multiple channels, although the presented modelling and development approach can be applied to services that will be delivered through a single channel only.

We should note here that in some cases multi-channel characteristics are needed to effectively deliver a service through a single channel. This situation may occur if the capabilities of the client devices used to access the service differ significantly. For example, in the case that a service involving fifteen fields is delivered through the web, these fields may be conveniently placed on a single form, if the client device will be a PC with screen resolution 1024x768; however multiple forms might be needed if the client device were a handheld computer with a screen resolution of 240x320 pixels. Similar considerations may hold for the input device capabilities (e.g. a limited keyboard makes the choice from a closed list more appropriate than typing,

such as the case reported for ISDN card phones [15] or mobile phones [16]), support of active features (e.g. JavaScript and particular versions of it) and so forth.
An additional incentive for organizations to adopt multi-channel service delivery (besides lifting the user-related barriers listed above) is that multi-channel delivery bears a number of benefits for the delivering organisation [17].
A particularly important reason for limiting the service delivery channels to the web can be traced to the costs associated with developing and maintaining multiple versions of the same service. Indeed, the development of a service that will be delivered through multiple channels is usually addressed as a number of distinct projects, with each project developing the service for a specific channel. Each such project is undertaken by a different development team (with channel-specific expertise), thus the service development cost increases significantly when multiple dissemination channels are used. Similar issues are faced with the maintenance procedure, where modifications to the service (due to legislation changes, modification of procedures or feedback from the service users) must be reflected to each distinct implementation. This approach is illustrated in Figure 1.

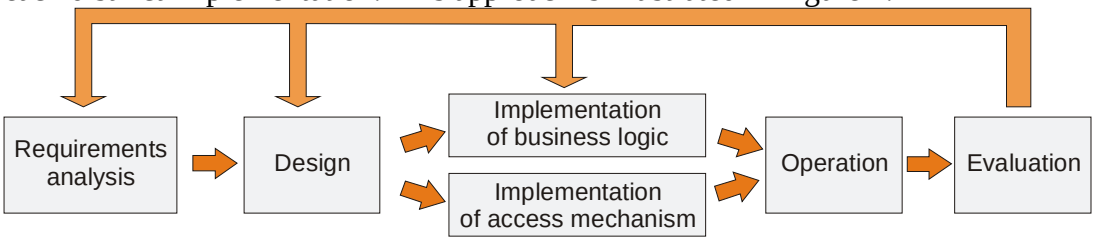

Figure 1 – Service delivery phases, repeated for each dissemination channel

Note that for an optimal delivery of electronic government services, structural reforms, and the adoption of a customer-centric model are required [18], which incur additional costs; these costs, however, are (largely) independent of the number of channels through which services are disseminated and will be undertaken by the organization even if only a single channel is employed.
In this paper we present a knowledge-based approach for developing and maintaining electronic services, which enables the automated production of service versions that are tailored for specific delivery channels. According to this approach, the (channel-independent) *business logic* is separated from the (channel-dependent) presentation issues for each delivery channel, allowing for the modelling the business logic to be performed only once. Finally, *generation engines* are employed to produce executable service images for the selected delivery channels.
Through re-using the modelling of the services' business logic and automating the error-prone task of platform-specific code generation, this approach minimises the time and resources needed for multi-channel service development. Maintenance activities are also facilitated, since changes to the services' business logic need to be performed only once, without the risk of inconsistencies. The approach has been complemented with appropriate tools, which enable the electronic service stakeholders to perform the related tasks, i.e. modelling of the service's business logic, creation of the channel-specific presentation schemes and, finally, the generation of executable service images.
The rest of this paper is organised as follows: section 2 surveys related work regarding multi-channel delivery of electronic services. Section 3 describes the modelling approach, while section 4 presents the related tools. Finally, section 5 concludes the paper and outlines future work.

2. Related work

The need for multi-channel service delivery has been recognised by the research community, industry and standards organisation alike. [19] attempts to determine the feasibility of a component-based approach, which will tackle the issue of highly fragmented ICT-architecture. This work argues that current ICT installations and know-how has been vertically organized around departments with little or not at all common horizontal functionality, hindering thus the development of multi-channel services. A number of e-government architectures that have been proposed insofar, namely the European one-stop government (EOSG [20], [21]), the Federal Enterprise Architecture (FEA [22]) and the extended Enterprise Application Integration ([23], [24]) include multi-channel access to e-services as core functionality of the delivery platform. Finally, in [25] a more holistic view for service access is undertaken, attending to the specifics of users, uses, organizational capacity, data characteristics, and technology, surveying among other features the multi-channel delivery of services.

In the industry sector, numerous software products have emerged in the past few years that facilitate the provision of services through multiple channels. The eGain service suite [26] incorporates a web, email, fax, letter management, and a contact centre, however the actual services are only provided through the web channel using the web self-service module and other channels play only a supportive role for communication and limited browsing. The same approach is also followed by Oracle Multichannel Customer Service [27], which integrates the web-self service module into the organisational service workflow.

Another approach that has been employed to accommodate more dissemination channels in service delivery with minimal development is the use of *gateways*. Under this approach, a service is developed and deployed for some *primary channel* (e.g. the web channel) and *gateways* are attached to the remaining channels (e.g. WAP, i-mode etc). When a gateway receives a request from the channel it is attached to, it arranges to retrieve the relevant content as encoded for the primary channel and employs a *dynamic translator* to re-encode this content into a form suitable for the channel it is attached to. Examples of such converters exist both in commercial products (e.g. [28], [29]) and in the open-source/shareware domain ([30], [31]). XSLT technology [32] can also be used to dynamically formulate content targeted to a specific dissemination channel, by providing an appropriate *transformation description*, which will be applied by the gateways to some XML-compatible representation of the content. This representation may be either tailored to some specific delivery channel (e.g. the web), or be written in a more “abstract”, semantics-oriented form (e.g. XML). Regardless of the specific technique used, however, gateways do not satisfactorily address the issue of diverse capabilities of client access devices, either within the same or across different dissemination channel, while their dynamic nature introduces processing overhead, which can prove overwhelming for heavily used sites.

In the standards area, XForms [33] is a considerable development that may be used to some extent for supporting multi-channel services. XForms provide a new platform-independent mark-up language for online interaction between an XForms Processor and a remote user agent. The basic feature of XForms that facilitates multi-channel service delivery is the separation of the *model* (form logic) from the *view* (channel/device dependent presentation), a provision that is in line with the approach directions outlined in section 1.This separation enables the client access devices to use the most appropriate controls to render the form, augmenting thus multiple device

support and accessibility [34]. The extent however to which an XForms-based application can be customised remains limited, since for instance placement of elements on forms is rigidly defined and cannot be dynamically formulated based on the channel and/or client access device characteristics. Another issue hindering the adoption of the XForms standard is the lack of support for it within the Internet browsing program community. Indeed, only recently beta releases of software for incorporating XForms support into the Internet Explorer and Mozzila/Firefox browsers have been announced [33], although working drafts and test suites for XForms have been available for more than two years. Some other implementations have been released in the past, but have proven unstable and/or with very limited functionality. Server-side implementations of XForms also exist (e.g. [35]), where the XForms model is stored on the server but it is translated to an appropriate representation *before* it is sent to the client, but still customisation capabilities remain limited.

3. A knowledge-based approach for modelling multi-channel electronic services

The proposed electronic service modelling approach uses high levels of abstraction for describing electronic service aspects rather than focusing on the software elements used for implementing them. Service modelling under this approach is divided into five stages, four of which, namely modelling of service purpose, content, business logic and evaluation criteria are channel-independent and need be to performed only once for each service, whereas the fifth stage, namely modelling of user access and visual layouts, is channel-dependent and needs to be performed for every delivery channel. These stages are described in the following paragraphs.

3.1 Modelling service purpose

To start with, each service has a specific *purpose*, e.g. it issues birth certificates or it allows for electronic submissions of tax return forms. The electronic service purpose normally stems from the organisation's mission statement and/or its operational framework. Typically, this stage is performed by managers, who have a high-level view of the organisation's mission statement.

3.2 Modelling service content

Another aspect of the electronic service that needs to be modelled is the actual *service content*, usually defined via a regulatory framework (laws, directives, regulations etc). This regulatory framework can be exploited in the context of describing an electronic service in three ways:

1. *the actual data that need to be collected by the service are defined*. For instance, the regulations pertaining to a birth certificate service may state that the requesting citizen should present some *proof of identity* (e.g. an identity card, a passport, a driver's license etc) and state his/her current address of residence. This implicitly states that an electronic service (or even a paper-based one) should provide the service user with some space to fill in the identity card number (or passport/social security number and so on) and some space in

which the residence address will be provided. For more complex services (e.g. tax return forms), the list of inputs may extend to a few hundreds of input fields. Each piece of data is usually complemented with some *metadata*, defining the field's description, labels for presentation etc. For the electronic versions of services, in particular, certain requirements may originate from the *electronic nature* of the service, e.g. requirement for presentation of a user name and a password, which will be given to the user after a registration process etc.

2. *validation checks that need to be performed for the user input*. In the simplest case, such checks will designate *mandatory items* that must be entered, as opposed to *provisional items*. For instance, providing a value for *the proof of identity* field is mandatory in a birth certificate electronic service, while entering the zip code of the address can be considered to be provisional. Type and range checks are another class of validation checks that should be modelled. Note that such checks may not always be expressly included within the regulatory framework but are known to domain experts, constituting thus *tacit knowledge* [36]. For instance, the regulatory framework of the tax return form submission may not expressly state that the number of children in the value entered in the *expenses* field should be a positive real number with two decimal digits, but domain experts do know that such a restriction should hold. Finally, another class of validation checks may involve multiple input fields within a single validation check, such as "if a value is provided for field A, then a value should be provided for field B as well" (e.g. if expenses for house rental are declared the landlord VAT number should be provided as well), algebraic relationships between values in different fields (e.g. the gross profit minus expenditures should equal the net profit, the interests cannot exceed the 10% of the capital etc). Again, such validation checks may either be stated within the service's regulatory framework or exist as tacit knowledge possessed by domain experts.
3. *explanations, examples and instructions.* Regulatory frameworks often contain detailed explanations and examples regarding the service content, especially for complicated items and notions (e.g. which expenses are considered deductible, how is the interest for overdue payments computed). Often, clarification documents are issued by administrations in order to give instructions, explanations or examples both to service end-users and to public authority employees, in regard to how certain aspects of the service should be handled. All these informational items can actually constitute parts of an electronic service, playing the role of *help items* or *detailed help items* that are available to users upon request. A help item may be associated with a single field (e.g. if a field accepts a VAT registration number, it may be associated with a help item exemplifying the required format of the VAT registration number), a *group of fields* (e.g. describing how an address field should be filled in) or even a validation check (for instance, exemplifying how values should be entered so as not to clash with the validation check).

Since the regulatory framework of electronic services is best known to the delivering organisation's domain experts, they are considered as the most suitable stakeholders to provide the information regarding the service content to the development process. Note again that all these concepts are expressed in a high level of abstraction, thus domain experts can directly enter the relevant information to the development platform, without any intervention from IT staff.

We point out that the *high level of abstraction* is coupled with *strict rules for concept expression*: e.g., the fact that some data item is mandatory is designated through an

appropriate check box, not by some verbal expression; similarly, appropriate user interfaces have been defined for entering validation checks [37], which are then translated into XML specifications [38], [39]. For instance, in order to express that the value of field A should be greater that the value of field B, the stakeholder first selects that "a field comparison validation check" is required, and subsequently selects "Field A" from a list presenting all available fields; afterwards the comparison operator "greater than" is selected, and the definition is finalized by selecting "Field B" from a list of fields. These rules for concept expression guarantee that the service specification will be free of vagueness or ambiguity, and can thus be mechanically processed to automatically generate code, as discussed in section 4.

3.3 Modelling service business logic

Once the *service purpose* and the *service content* have been defined, the service's business logic can be complemented by defining the *processing* that the submitted data will undergo. This task is normally undertaken by staff with IT expertise within the organisation, or outsourced to IT specialists. Specification of the processing may be performed using one of the following two methods:

1. direct provision of the code that will process the data. Under this scheme, the IT staff supplies code that will be invoked whenever a new submission is made through the electronic service. Upon invocation, the code is provided with an XML document containing all the data submitted by the user *plus* meta-information regarding the submission, including the credentials used by the user to validate him/herself to the submission platform, the time that the submission was initiated and the time the submission was concluded, a pointer to the XML schema that the document adheres to and so forth. The code should extract the data from the XML document, process it accordingly (possibly including interaction with organizational repositories, execution of external programs or other pertinent actions) and return a specifically formatted XML document, containing an indication whether processing was successful or not, and a reply to the user, in HTML format. This option corresponds to *synchronous request processing*.
2. incorporation of the submission into the organizational workflow. In many cases submissions cannot be directly processed for a number of reasons (physical inspections may be required, production of physical objects –such as passports– may be necessary, formulation of the reply may be computationally expensive, etc). In these cases, the submitted data may be incorporated into the organizational workflow, from where it will be extracted and processed -similarly to requests arriving from "traditional" channels, such as the counter. Incorporation of the request into the organizational workflow is typically performed by inserting the relevant data into a repository (database) or by executing an external program (part of the workflow management system) which should be provided with a suitably formatted file containing the task information. In these cases, the IT staff should provide the code that will populate the repository or place the submission data into a file and invoke the appropriate program. Again, this code is invoked upon each submission to the service and is provided with an XML document containing submission data and meta-information, as in the previous case.

For more information on the data processing specification scheme, the interested reader is referred to [40]. Due to its code-oriented nature, this part of the service

modelling and development process has no visual interface and thus has no corresponding module within the platform presented in the following section.

3.4 Modelling user access and visual layouts

The next step for completing the modelling of a service that will be delivered to multiple platforms is the specification of the service delivery environment (e.g. Web, WAP etc) and the provision of the service visual layout for each of the platforms. It is possible to define *multiple visual layouts* for a single service delivery channel, in order to cater for client access devices with radically different capabilities (e.g. screen resolutions, input methods and so forth).
Effectively, a visual layout is a *set of forms* in a representation appropriate for the channel to which the form set pertains (for instance HTML/XHTML for the web, WML for WAP, cHTML for i-mode and so forth). The visual layouts are *linked* to elements of the service content (fields and help items) by injecting proper *mark-up tags* within the form representation.

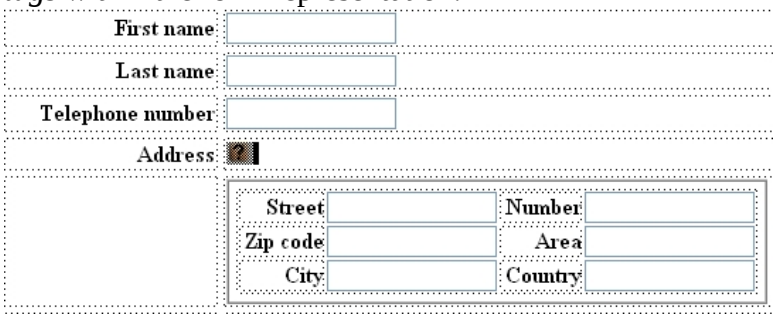

Figure 2 – Selecting the visual layout component to be linked

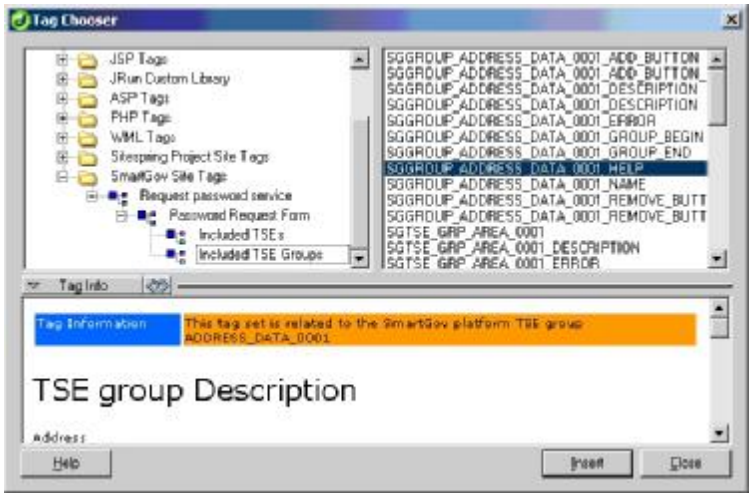

Figure 3 - Selecting the service content item to be linked to the visual layout component

The procedure of mark-up injection can be either performed manually, or by means of suitably extended visual editing tools, e.g. Macromedia DreamWeaver™. Figure 2 and Figure 3 illustrate how such links may be established in the environment of Macromedia DreamWeaver MX™. In Figure 2, the user selects the visual layout to be linked, while in Figure 3 the user employs the *tag chooser tool* of DreamWeaver to select the corresponding element of the service content. The list of service content elements presented within the tag chooser tool is automatically created by a tool that

retrieves the descriptions of service content elements and formats them suitably for DreamWeaver MX™.
Note that only field-type and instruction/example-type elements may be linked through this procedure: validation checks are automatically associated with the form on which the pertinent fields appear and are conducted when the form is submitted (if the client environment permits, some checks can be also conducted upon entering the field value, for example by using JavaScript in a browser). If the pertinent fields appear on different forms, then the validation check is automatically associated with the *service as a whole*, and they are conducted upon the *final submission*, when all field values are available. The code generation module uses these mark-ups to insert appropriate widgets and/or links for the service content elements, taking to account the requirements of the service dissemination channel and the form specification language that is used. The algorithm used for generating executable versions of the service based on the service description is presented in section 4.

3.5 Modelling of the service evaluation criteria and procedures

Each service is also evaluated using specific *criteria*, which can be quantified via *statistical measures*. For example, the impact of the service can be quantified via the daily number of service users; service response time can be measured in absolute terms by computing the time between the service invocation and the delivery of the result to the service user, while the perceived service response time can be derived from the service users' replies to a specific questionnaire item "Please rate the service responsiveness (1=low, 9=high)". These criteria and statistical measures will be used by managers to assess various aspects of the service. Note that the abstraction level of these information items is adequately high to enable managers and domain experts to directly specify them to a modelling environment without any intermediation of systems analysts or IT personnel.

4. The knowledge-based multi-channel electronic services modelling platform

The tools necessary for supporting the methodological steps described in the previous section have been implemented in the context of the SmartGov project [41] as an integrated platform for developing e-government services. The overall platform architecture is illustrated in Figure 4. Electronic service development stakeholders use the six first modules (service purpose definition, field definition, validation checks definition, instructions definition, visual element definition and evaluation metric definition) to describe the relevant aspects of the service. The visual elements definition module is employed to provide the appropriate service visual layout per target delivery channel and client device access capability set. Besides the "service purpose definition", which should commence first, no sequence requirements are imposed for the other modules, e.g. it is not required to complete the definition of all fields before moving on to defining validation checks or instructions. Any validation check can be defined as soon as the fields involved in it have been defined, whereas instructions and help texts can be created even before the items they pertain to have been defined, and may be linked to them afterwards.

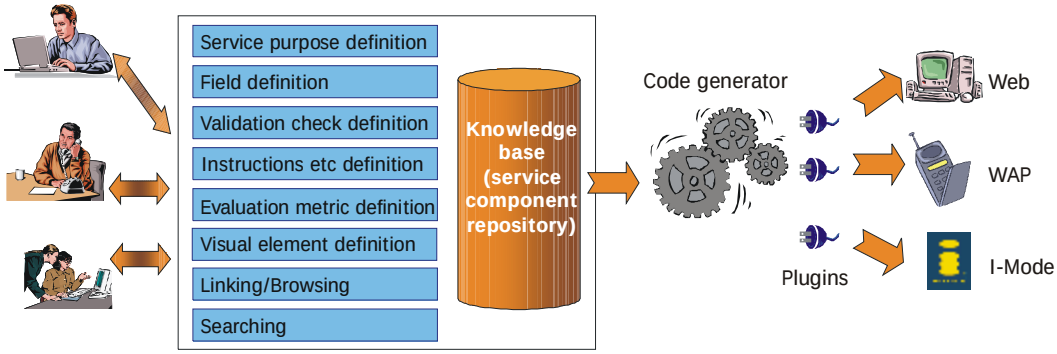

Figure 4 - Overall platform architecture

The service development procedure starts off by employing the "service purpose definition" module, illustrated in Figure 5. Besides the purpose of the service a number of other parameters may be entered here, including the deadline for submitting documents through the service (if applicable), whether service users are allowed to delete or edit already submitted documents, authentication requirements etc.

Figure 5 – Service purpose definition module

Once the service purpose is defined, service content definition may commence, employing the field definition, validation check definition and instructions and help texts definition modules.

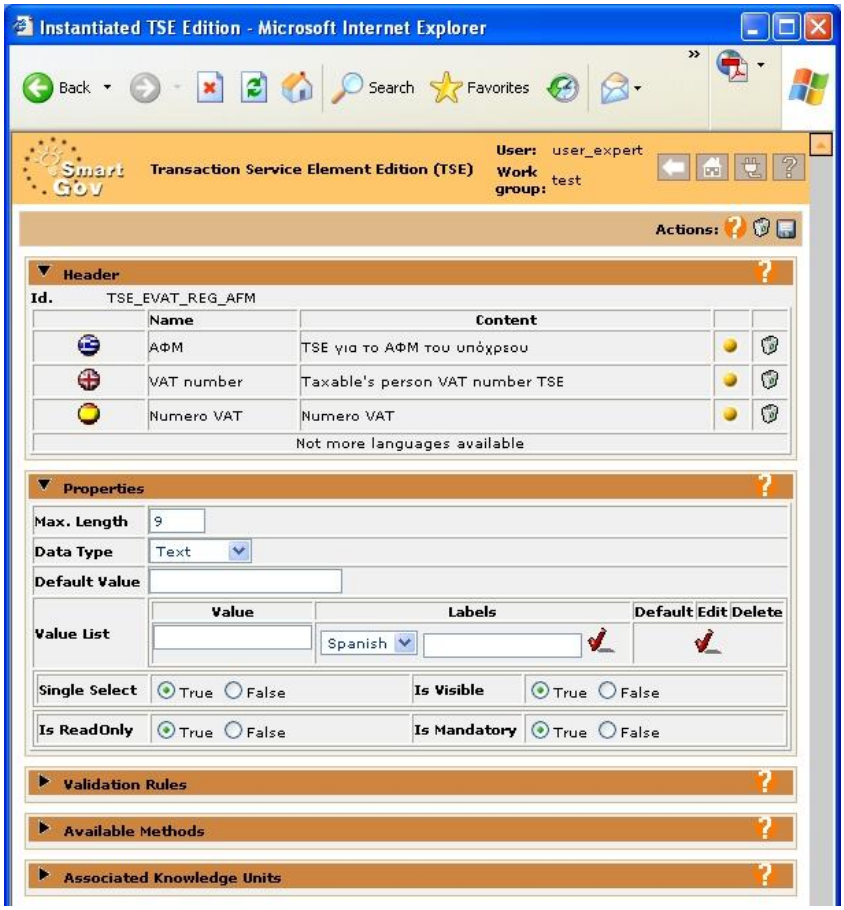

Figure 6 - Field definition module

The field definition module (Figure 6) allows the domain experts to define the data that need to be collected by the service, entering all pertinent parameters, including multilingual labels, basic properties of the user input (e.g. whether texts, numeric inputs, dates etc are expected, whether input is mandatory or not, if the values are for display only or the user may modify them and so forth).

User input is further verified using *validation checks*, which are defined through the validation check module (Figure 7). In this screenshot we may see that for a validation check the user should provide a description, the message that should appear if the condition is violated and the condition itself. A number of standard condition types are available modelling the most common checks, while for more complex ones the ability to use a specifically designed macro language or even native code (Java and JavaScript) is provided. If a validation check references a specific field, then it automatically appears under the "Validation rules" section in the field definition module (Figure 6) when the specific field is edited, allowing domain experts to easily view which validation rules are applied to each piece of data.

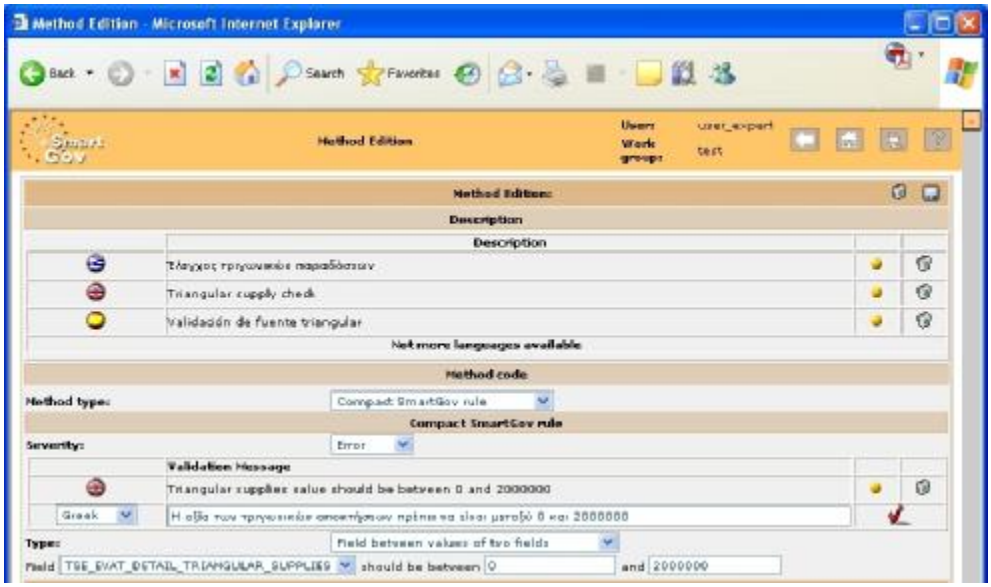

Figure 7 - Validation check definition module

The third task within the content definition phase is the specification of instruction and documentation for the electronic service. This task is facilitated by the instructions module (Figure 8), which permits domain experts to enter any information necessary for (a) helping the end-users to efficiently use the service and (b) internally document the project. To this end, when a domain expert enters some piece of instructions through the instructions module, he/she specifies the user groups to which it is addressed (managers, domain experts, IT staff, end-users). Only instructions listed as addressed to end-users may be included in the running version of the service; service development stakeholders may view a piece of instructions if it is addressed to the specific stakeholder group. Typical items that are entered as project documentation but are not made available to service users include the legislation on which the service or portions of it are based, internal evaluations or quality check procedures, instructions for other service stakeholders that will re-use a specific component in another project and so on.

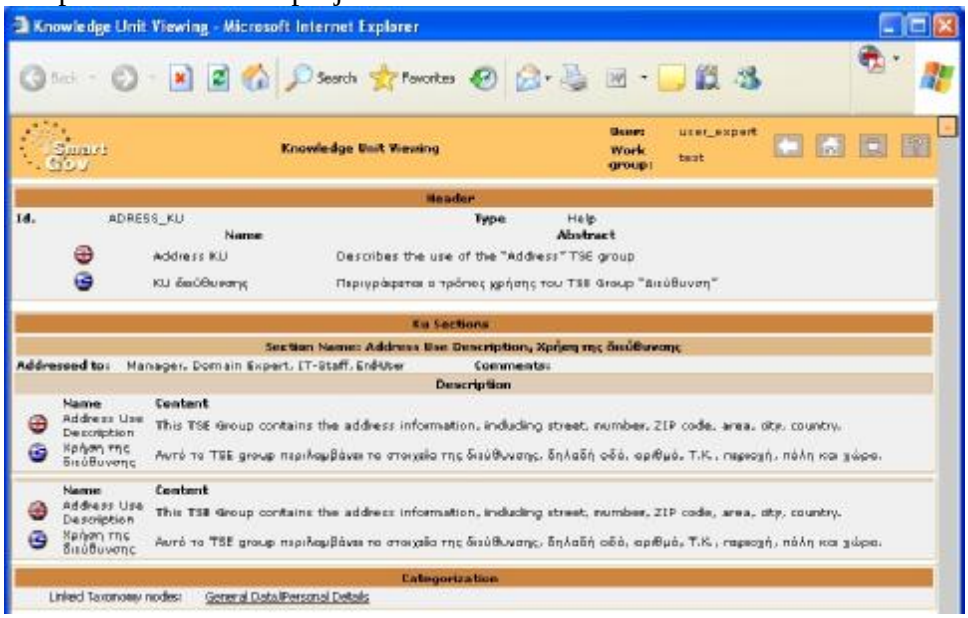

Figure 8- Instructions definition module

Each piece of instructions may be associated with one or more elements (fields, validation checks, visual layouts etc) – for instance, a piece of instructions describing how addresses should be entered will be linked to all forms and/or fields related to addresses, which may belong to different electronic services.
The content specified using the previous three modules must then be associated with visual layouts, one for each intended dissemination channel and/or client device capability set. This can be accomplished through the visual element definition module (Figure 9), where the target dissemination platforms are defined and for each one, a set of forms (prepared as described in section 3.4) is defined. In the example presented in Figure 9, two target dissemination channels are defined, namely the web channel and i-mode. The web channel employs only two forms (header and detail), whereas the i-mode channel needs three (the second one has been split into two parts) since the client devices used for accessing it typically have smaller screens. Moreover, the first form for the i-mode channel is "compacted" i.e. only the absolutely necessary information is included to save bandwidth. This provision does not only apply to screen dimensions, but may include content types, quality or any other aspect that may be differentiated across platforms. For instance, help videos may be provided for interactive TV and broadband internet connections, but not be available through WAP; audible instructions may be given for phone centres and interactive TV, whereas equivalent textual representations may be used for WAP and web; The forms used for the web channel may include artwork for aesthetic purposes, which may be missing from other channels so as to improve access time and so forth. In this respect, the modelling scheme allows to tailor the interface for each specific dissemination channel to use the features and content considered as "most appropriate".

Figure 9 – Visual element definition module

The final element that should be defined for services is the evaluation metrics and statistics. These are specified using the corresponding module. The platform offers a number of pre-defined metrics which can be enabled or disabled (Figure 10), while a sub-module enabling definition and deployment of questionnaires is also available. Some technical statistics are also available, such as execution time for validation rules, memory consumption etc, enabling IT staff to profile and optimize the service.

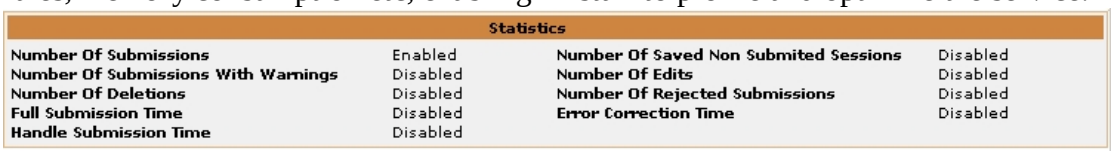

Figure 10 – Evaluation metric definition module

All the elements defined by the transaction service development stakeholders are stored into the *knowledge base*, which is effectively a service component repository complemented with a network of semantic links interconnecting the different elements. Element linkage may be performed automatically (as is the case of validation checks being linked to the fields they are associated with), semi-automatically (the system may propose instructions and documentation to be linked to fields, based on textual similarity of descriptions and content) or manually (e.g. fields are linked manually to the visual layouts, as explained in section 3.4). The link network can be exploited in a number of ways. First, it allows transaction service

stakeholders to navigate between interrelated components during the development process and thus easily locate elements of interest. Second, it promotes consistency by allowing the system to perform a number of checks; if, for instance, a field is deleted, then the system may raise an alert notifying the stakeholder that the validation check has a dangling field reference. Another important use of the link network is to help identify elements that need to undergo maintenance activities due to legislation changes: if a piece of legislation has been modified, then the respective documentation item is located and the links from it are exploited to locate components that are based on it. These fields can then be modified in accordance to the updated legislation.

The platform offers two more means to transaction service stakeholders for locating components, namely taxonomies and searching. Taxonomies are used to classify elements according to pre-determined systems, and stakeholders subsequently traverse these classifications to locate elements specific to a specific branch of the classification. The platform presented in this paper allows for multiple taxonomies to be used (e.g. in Figure 11 two taxonomies are defined, one for taxation and one for customs), and any element may be linked to any number of taxonomies and/or in multiple branches of the same taxonomy. This is useful for elements used in multiple contexts, e.g. the "Tax payer's personal details" element is used in all services related to taxation and all services related to customs.

The search mechanisms incorporated into the platform accept search patterns that include free text search, specification of whether the match is required to be within a specific attribute of the element (label, description, documentation, validation rules etc) or in any of them and the element type of the result (field, purpose, visual layout etc). Another option of the search mechanism is to extend the search to *directly linked components*, to address the issue of information being scattered between various elements. For instance, if the user enters as a search criterion the string *surname address phone*, there may not be a single element containing these words, but a personal detail form contains direct links to fields whose texts *collectively* satisfy the search criterion. The ability to locate existing components has proven to be valuable for promoting re-usage of already developed components in new services [42].

Figure 11 – Using taxonomies for organising and locating components

Once all aspects of a service have been defined, the *code generator* module is executed to automatically produce the executable service images for the various platforms. The code generator module extracts the pertinent definitions from the repository, and then employs appropriate *plug-ins* to produce platform-specific code for each element. A separate plug-in is required for each target dissemination channel, e.g. a web code generator plug-in is needed to produce executable service images for

the web dissemination channel, a WAP code generation plug-in will create executable service images for the WAP channel and so forth. Each plug-in encompasses the knowledge on how the high-level constructs of the service repository components will be translated into corresponding low-level elements, as required by the platform. For example, in the web environment, each user interaction form will constitute a different page, and the user will move between the pages using *<input type="submit">* form elements; in the WAP environment, forms will be modelled as separate *cards* within a *deck*, and the navigational controls between the pages will be implemented using a *<do type="accept" label="Next form">* tag, with an appropriate *<go>* tag content. Similarly, modelling active behaviour within the client access device (this includes validation checks and confirmation dialogs with the user) will be coded in JavaScript for the web environment and in WMLScript [43].

In brief, the executable service image generation algorithm proceeds as follows:

1. For each *visual description* of the service, the appropriate target channel is selected (as indicated by the stakeholder that provided the visual description) and the relevant code generator plug-in is loaded.
2. For each form within the visual description, the appropriate element is created (e.g. HTML page, WML card) and elements that are linked to the form (through the tag injection procedure) are fetched from the repository. Input elements are translated to controls suitable for the platform and navigation controls enabling the service user to move between the forms are inserted. The element is finalised by inserting active behaviour modelling the validation checks that involve *only* input elements belonging to the form being processed. Validation checks involving elements on different forms are handled separately (explained below).

In addition to the code described above which is sent to the client access devices, the code generator additionally emits code that runs on the *server side* while the service is being accessed. This code encompasses the following functionality:

1. intercepts the form submissions made by the user and performs the validation checks on the provided data. Re-checking the values at the server is considered compulsory, since clients are considered untrustworthy (e.g. a web browser may have the JavaScript engine disabled or a malicious client may deliberately submit erroneous values).
2. arranges so that the correct client code stub is sent to the client access device when navigation activities take place.
3. runs validation checks for fields that appear on different forms. This action is associated with the event of the “final submission” (triggered by selecting an appropriate button on the last form), since all input values are available at this stage. If any of these validation checks fail, the final submission is not accepted and the user is prompted to correct the given values.
4. provides *hooks* for the IT staff to integrate custom code that will run upon invocation of the service by a user and code that will run after the final submission (provided that all validation checks have been successful). Typically, startup code will validate user credentials (such as user names and passwords) and will fetch from organisational repositories data that need to appear as pre-filled in (e.g. once a user has been authenticated via the user name and the password, the user’s personal details [name, surname, address, etc] can be retrieved from such a repository); code that runs after the final submission usually arranges for storing the submitted values to organisational

repositories, so as to be processed by the appropriate organisational workflow chain.

The server-side code emitted by the code generation is bundled in appropriate JSP files, which are bundled along with client-side files (generated files plus images and stylesheets, if applicable) and runtime libraries into a WAR package, suitable for deployment to a JSP container, such as Tomcat [44] and BEA Weblogic [45]. The WAR file should then be deployed to a JSP container that is connected to the pertinent dissemination channel. Note that in the current version of the platform, the case that multiple versions of the same service have been produced for the same channel (e.g. two versions for the web channel, one targeted to PC-based environments with high screen resolutions and one targeted to PDAs with small screen resolutions), these versions will be deployed as *separate services*, without any linkage between them or directions to the users. Site administrators are expected to create a (trivial) initial page for the service that will present the existing alternatives so that the users may choose. More sophisticated versions of such pages may auto-detect the capabilities of the client access device and select transparently the service version that best matches the profile of the client access device.

5. Conclusions

In this paper we have presented a knowledge-based approach for developing e-government services, which can be delivered through multiple dissemination channels. This approach separates the service's business logic from the presentation issues, avoiding thus duplication of work that will be incurred if each delivery channel is addressed in isolation. Using high-levels of abstraction, the approach facilitates the direct modelling of services by managers and domain experts, minimising thus the involvement of IT staff, which is usually a scarce resource in organisations. The modelling approach is complemented with a development environment in which electronic services can be modelled and a code generator, which reads the service models and automatically produces executable server images for the pertinent platforms. Future work will focus on the integration of multiple services into *life events*, adopting thus a citizen-centric model for the platform. Reverse engineering of existing electronic services into the knowledge-based platform in order to minimise both the service maintenance costs and enable the multi-channel delivery of existing services with the minimal possible cost will be also investigated. Finally, the automatic production of *initial pages* that will guide users to the correct version of the electronic service and the provision of facilities to use multiple channels within the same *service session* (e.g. completing some forms using an i-mode browser and then switching to a PC-based environment to complete the work) will be addressed.

References

[1] European Commission, 1999. Public Sector Information: A Key Resource for Europe, Green paper on Public Sector Information in the Information Society. http://europa.eu.int/ISPO/docs/policy/docs/COM(98)585/

[2] Gil-Garcia Ramon 2004. Electronic Government in ISWorld Encyclopedia. Accessible at http://ispedia.terry.uga.edu/?title=Electronic_Government [Accessed February 2005]

[3] e-Europe, 2000. Common list of basic public services. Accessible at http://europa.eu.int/information_society/eeurope/2002/news_library/documents/basicpublicservices.doc [Accessed February, 2005]

[4] Cap Gemini Ernst & Young, 2003. Online availability of public services: How is Europe Progressing?
[5] Cap Gemini Ernst & Young, 2004. Online availability of public services: How is Europe Progressing? Available at http://europa.eu.int/information_society/eeurope/2005/doc/highlights/whats_new/capgemini4.pdf [Accessed February, 2005]
[6] United Nations, 2003. World public sector report 2003: e-government at the crossroads available at http://unpan1.un.org/intradoc/groups/public/documents/un/unpan012733.pdf [Accessed February, 2005]
[7] Internet World Stats, 2005. World Internet Usage And Population Statistics. Available at http://www.internetworldstats.com/stats.htm [Accessed February, 2005]
[8] Ni, Ya, A., Ho, Tat-Kei, A., 2005. Challenges in e-government development: Lessons from two information kiosk projects, Government Information Quaterly vol. 22, pp. 58–74
[9] Slack F. and Rowley J., 2002. Kiosks 21: a new role for information kiosks? International Journal of Information Management vol. 22, pp. 67-83
[10] Nurmela J., Heinonen R., Ollila P., Virtanen V., 2000. Mobile Phones and Computer as Parts of Everyday Life in Finland. Available at http://www.stat.fi/tk/el/stty2r1e.html [Accessed February, 2005]
[11] E-Business Forum, 2004. Basic indicators of ICT usage in Greece. Available at http://www.ebusinessforum.gr/index.php?op=modload&modname=Sitemap&action=sitemapviewpage&pageid=149&language=en [Accessed February, 2005]
[12] General Secretariat for Information Systems of the Greek Ministry of Finance, 2003. TaxisPhone usage statistics. Available at http://194.219.152.68/statistika/phone/statist.htm [in Greek, accessed February 2005]
[13] General Secretariat for Information Systems of the Greek Ministry of Finance, 2004. Tax return form electronic submission. Available at http://www.e-oikonomia.gr/allframes.htm [in Greek, accessed February 2005]
[14] Interchange of Data between Administrations, 2004. Multi-channel delivery of eGovernment services. Available at http://www.cisco.com/global/DE/pdfs/publicsector/ida_07_04.pdf [Accessed January, 2006]
[15] Maroulis D., Aronis S., Nassiopoulou V., Grammenos N., Vassilakis C., 2004. A Web Browser for ISDN Card Phones, Journal of Internet Technology, Vol 5., No 3
[16] Silfverberg, M., 2003. Using mobile keypads with limited visual feedback: Implications to handheld and wearabledevices. In Proceedings of Mobile HCI 2003, pp. 76–90
[17] Hiles Terry, 2004. The Multi-channel Service Revolution. Available from http://www.ebusinessforum.gr/index.php?op=modload&modname=Downloads&action=downloadsview&pageid=1113 [Accessed February, 2005]
[18] Robben F., 2001. (Re)-organising for better services, eGovernment Conference: From Policy to Practice, 29-30 November 2001, Charlemagne, Brussels. http://europa.eu.int/information_society/eeurope/egovconf/documents/ppt/Annex_2_Benchmarking_Robben_presentation_30-11-2001.ppt
[19] Janssen, M., Wagenaar, R., Beerens, J. Towards a Flexible ICT-Architecture for Multi-Channel E-Government Service Provisioning. Proceedings of the 36th Annual Hawaii International Conference on System Sciences, (HICSS'03), 6-9 January 2003, http://csdl2.computer.org/comp/proceedings/hicss/2003/1874/05/187450148.pdf
[20] Wimmer, M., Krenner, J., 2001. An Integrated Online One-Stop Government Platform : The eGOV Project, 9th Interdisciplinary Information Management Talks, Proceedings, Schriftenreihe Informatik, Universitätsverlag Trauner, Linz, pp.329-337.
[21] Tambouris, E., 2001. An Integrated platform for Realising Online One-Stop Government: The eGOV Project. Proceedings of the DEXA International Workshop "On the Way to Electronic Government", IEEE Computer Society, Press, Los Alamitos, CA.
[22] U.S. Office of Management and Budget, 2005. Federal Enterprise Architecture. http://www.whitehouse.gov/omb/egov/a-1-fea.html [Accessed February, 2005]
[23] Avignon, L., Brethes, T., Devaux, C., Pezziardi, P., 1999. Le livre blanc de l'EAI I'ntégration des Applications d'Entreprise., Livre blanc, Octo Technology, October 1999.
[24] Mediadev, 2002. Pourquoi et comment faire communiquer les applications. Livre blanc, Mediadev.
[25] Dawes, S., Pardo, T., Cresswell, A., 2002. Designing Government Information Access Programs: A Holistic Approach. Proceedings of the 36th Hawaii International Conference on System

Sciences (HICSS'03), IEEE Computer Society Press, http://csdl.computer.org/comp/proceedings/hicss/2003/1874/05/187450146a.pdf
[26] eGain, 2004. Service Suite product presentation. Available at http://www.egain.com/products/multichannel_service.asp [Accessed February, 2005]
[27] Oracle Corporation, 2004. Oracle Multichannel Customer Service FAQ. Available at http://www.oracle.com/applications/service/faq_mch_1201.pdf [Accessed February, 2005]
[28] Oracle Corporation, 2002. Oracle9iAS Wireless Developer's Guide Release 2 (9.0.2).
[29] IBM Corporation. WebSphere® Transcoding Publisher product description. Available at http://www.ibm.com/software/pervasive/transcoding_publisher/ [Accessed February, 2005]
[30] Sourceforge team, 2004. Dynamic HTML Conversion to WML. Available at http://html2wml.sourceforge.net/ [Accessed February, 2005]
[31] Akerman Software, 2004. Mobile Converter v. 1.0.3. http://www.akermansoftware.com/ [Accessed February, 2005]
[32] Mangano S, 2002. XSLT Cookbook. O' Reilly, ISBN: 0596003722
[33] W3 Consortium, 2003. XForms - The Next Generation of Web Forms. Available at http://www.w3.org/MarkUp/Forms [Accessed February 2005]
[34] Raman, T. V. 2003. XForms - XML Powered Web Forms. Available at http://safariexamples.informit.com/0321154991/book.html [Accessed February, 2005]
[35] FormFaces, 2004. FormFaces product description. Available at http://www.formfaces.com/ [Accessed February, 2005]
[36] Polanyi, M., 1998. The Tacit Dimension, In Prusak, L. (Ed.) Knowledge in Organization, Butterworth-Heinemann, Boston, MA.
[37] SmartGov Consortium, 2003. SmartGov Project Deliverable Deliverable D82: User's guide. Available from http://www.smartgov-project.org [Accessed February, 2005]
[38] SmartGov Consortium, 2003. SmartGov Project Deliverable D51-61: Low-level Specifications of SmartGov Services and Applications and the Knowledge-Based Core Platform. Available from http://www.smartgov-project.org [Accessed February, 2005]
[39] SmartGov Consortium, 2003. SmartGov Project Deliverable D52: Deliverable D52: Implementation of SmartGov Knowledge-Based Core Platform. Available from http://www.smartgov-project.org [Accessed February, 2005]
[40] Vassilakis C., Lepouras G., Rouvas S. and Georgiadis P. (2003). Integrating e-Government Public Transactional Services in the Public Authority Workflow. Electronic Government J., vol. 1, Inderscience Publications, pp. 49-60
[41] SmartGov Consortium, 2004. SmartGov Project Web Site, http://www.smartgov-project.org [Accessed January, 2006]
[42] Vassilakis, C., Lepouras, G., 2004. Reusability in Electronic Services Development. Proceedings of the CSITeA 04 conference, Cairo, Egypt.
[43] Frost Martin, 2000. Learning WML & WMLScript. O'Reilly, 1st edition, ISBN: 1565929470
[44] Brittain J., Darwin I. F., 2003. Tomcat: The Definitive Guide. O'Reilly, ISBN: 0596003188
[45] Zuffoletto J., Miranda L, 2003. BEA WebLogic Server Bible (second edition), Wiley, ISBN: 0764526022.