REUSABILITY IN ELECTRONIC SERVICES DEVELOPMENT

Costas Vassilakis
Department of Computer Science and Technology
University of Peloponnese
Terma Karaiskaki, 22100, Tripolis, Greece
costas@uop.gr

George Lepouras
Department of Computer Science and Technology
University of Peloponnese
Terma Karaiskaki, 22100, Tripolis, Greece
G.Lepouras@uop.gr

Abstract

Electronic government employs electronic services to facilitate interaction with citizens and enterprises and deliver a rich and high quality spectrum of services. Development of electronic services can be greatly assisted, both in terms of development cost and roll-out time, by exploiting the reusability inherent in them. Reusability may be promoted by identifying reusable objects in the context of electronic service development, building and populating a repository with such components and providing the means for developers to locate, extract and adapt them to suit the task at hand. In this paper we analyse electronic services to recognise reusable components and present means and techniques that empower electronic service developers to build electronic services through reusable components.

1 INTRODUCTION

Electronic services, especially transactional ones, are a central tool in electronic government, since a considerable number of services offered to the citizens or enterprises, in the context of electronic government, are modelled through such transactional services. It is worth noting that among the basic public services listed in [1], 15 of them (75%) are transactional services, i.e. services that involve filling-in and submission of electronic forms.

An electronic service is a complex software artefact, whose development requires the cooperation of numerous experts with diverse skills. Firstly, domain experts are needed who possess the know-how regarding the business processes that must be modelled and the rules that govern them. Secondly, analysts are needed who will interact with domain experts to extract the requirements for the electronic service. These requirements will then be passed to the IT staff who will implement not only the service logic and the code to enforce business rules, but connections to back-end repositories as well, regarding data storage and retrieval. Finally, the visual layout of the service needs to be refined and optimised by experts in computer-human interaction. During the maintenance phase, some tasks may be revisited to cater for accommodating changes in the environment (e.g. change of legislation, specifications or IT environment such as databases) or to improve the quality of the service provided.

The current practice for developing electronic services makes little or not at all benefit of the reusability concept: for each electronic service, all analysis, development and maintenance tasks are performed anew, even though the same or similar tasks have been performed in the context of other electronic services. This can be attributed to the fact that electronic services are mainly viewed from the business point of view, with respect to which each electronic service performs an altogether different function than other deployed electronic services, thus the opportunities to employ reusability are limited. From a software architect's point of view, however, an electronic service actually consists of smaller, simpler building blocks, that can be reused across electronic services in the same way that software libraries [2], object classes and packages [3] and patterns [4] can be reused for building software applications. Actually, in the context of electronic services reusability opportunities may be even higher than in traditional application development, since reuse may extend to requirements analysis – for example, if the "personal detail collection" requirement is analysed for a specific service that is based on some legislation, then this analysis (and all derived design and implementation) can be reused for other services based on the same legislation.

In this paper we propose an approach for electronic service development that promotes reusability. We first analyse electronic services to identify their basic building blocks and recognise the opportunities to employ reusability at various levels (single elements, element groups etc) and varying scopes (departmental, inter-departmental, inter-organisational and so on). We then propose means and techniques that will allow service developers to reuse existing components, or create their own and make them available for other developers to reuse. Our proposal is based around a *reusable component repository*, which is complemented with browsing and search facilities that enable developers to examine and query its content.

The rest of the paper is organised as follows: in section 2 the basic building blocks for electronic services are identified. Section 3 presents a repository-based electronic service development approach that promotes reusability and examines facilities that need to be available for this approach to be effective. Finally section 4 concludes and outlines future work.

2 ELECTRONIC SERVICE BUILDING BLOCKS FOR REUSABILITY

Electronic services are, generally, computerised equivalents of business processes involving filling in and submission of forms, processing of submitted forms and possibly return of a reply to the submitting citizen.

When using an electronic service, the user is presented with a *number of forms*, which must be filled in. Short documents may be represented using a single electronic form, whereas lengthy documents may be partitioned into multiple forms. A form may comprise of several *areas*, and each area commonly contains individual *fields*, which are conceptually interrelated. For example, in a tax return form distinct areas may be dedicated to collecting data regarding the taxpayer's personal details, income and expenditures. Form fields are the individual elements that citizens need to fill in, either by direct typing of data in the area pertaining to the field (e.g. typing *13765* in the input area of the *Zip code* field) or by selecting one of the available field options (e.g. *Yes* or *No* for the *Do you own the house you live in?* field). Fields usually come complete with *labels*, i.e. descriptions of their purpose on the form. In some cases, the number of fields needed for some purpose cannot be predetermined. For example, if the *Protected family members* need to be declared, the number of entries may vary from one (single person) to twenty or more, and for each one of them the name, the surname and the relationship to the declaring citizen must be declared, as shown in Figure 1.

Figure 1 – Repeating fields to accommodate input

In the context of electronic services, two extra facilities are available for fields, as compared to the paper-based versions: firstly, some fields may be automatically filled in by the system; for instance, if the user presents a username and a password to log into a service, the user's personal details may be retrieved from a registry and be automatically placed in the corresponding fields. Another facility is to automatically compute the contents of some fields, representing for example percentages of a value or column/row sums. In most cases, fields that are automatically filled in or computed are not allowed to be directly changed by the electronic service user.

While the above elements of an electronic service (forms, areas, fields) provide the required functionality to the end user, of equal importance are the *instructions* that are made available to the citizens, regarding the use of the electronic service. Instructions may contain explanations, examples, step-by-step guides, help desk contact details or any other material that will assist the citizens to use the electronic service. Typically, specific instructions are accessible via hyperlinks that are located close to the fields they pertain to; general instructions and examples covering field areas or whole forms may be anchored in a more global context, e.g. close to the top of the form or in a separate toolbox.

In addition to these components, which are targeted for use by the citizen accessing the electronic service, an electronic service normally encompasses a number of complementary elements that are counterparts of the back-office work that is associated with the modelled business process. One important part of this work is the conducting of *validation checks*, to ascertain that the forms are filled-in by the citizen in conformance to the instructions. Validation criteria may dictate that some elements are mandatory (e.g. the submitting citizen's surname must be filled-in), limit the value range that can be input within a single field (e.g. the text entered in the *Date of birth* field should represent a valid date, whereas the inputs in the *Gross income* field must be a positive number). The most complex type of validation criteria includes cross-checking of different fields or different forms (for example "if the net profits field is filled in then the net loss field should be left blank"; "the net profit cannot drop below the 30% of the gross profit"; "form A cannot be submitted before form B"). In paper-based environments, conformance of submitted documents with respect to validation checks is conducted by either front-desk workers receiving the document from the citizen, or by the back-office workers that will process the form. Validation checks that apply to a particular service stem from the relevant *legislation*, which also defines the purpose of the service, the format and content of the documents that must be submitted, the citizen classes that can submit the documents and the related submission periods, etc. In some cases, certain portions of legislation may affect multiple electronic services; for instance, the legislation defining the format of the VAT number affects all electronic services that include VAT numbers. Legislation needs to be related to electronic services and their components for a number of reasons, including reference by citizens and workers of PA, documentation of the service and tracking of elements affected by legislation changes.

Finally, when a form is submitted it needs to be *filed*, for future processing and reference. In the context of electronic services, filing is equivalent to storing the document in the appropriate repository of the organisational information system, in a form appropriate for further processing within the organisational workflow. Since the system delivering the electronic service to the citizens is usually separate from the main organisational system (due to both technological and security considerations), a communication scheme between the two systems must be established to fulfil this task.

Summarising the components of electronic services, the following building blocks may be identified:

1. forms
2. form areas
3. fields
4. instructions
5. validation checks
6. legislation
7. communication with back-office systems

Having identified the elementary electronic service building blocks, an issue that must be investigated is whether these building blocks can be directly used for the purpose of promoting reusability in the context of electronic services. The criteria that a component must meet in order to be considered as a reusable part may be extracted from [5], according to which "a component is a non-trivial, nearly independent, and replaceable part of a system that fulfils a clear function in the context of a well-defined architecture. A component conforms to and provides the physical realization of a set of interfaces". We will now review the extent to which the identified electronic service building blocks meet the component reusability criteria.

Firstly, *fields*, as defined above fail to meet the triviality criterion. Indeed, an input area coupled with a label, can be easily created by the electronic service developer, with a total effort less than the one needed to locate and extract the corresponding element from the reusable component repository. The triviality criterion is met by all other building blocks, although *some instances* of specific components may be trivial, but this cannot be generalised for the whole building block category (e.g. some instructions may simply be "type in a number", but *not all instructions* are that trivial).

Another important criterion that the identified building blocks fail to meet is that of the *near independence.* Indeed, validation checks cannot be viewed independently of the fields, form areas, forms or services they pertain to; a validation check can only be used in a context that the fields involved in the validation check exists. For example a validation check checking that the pre-paid taxes are less than 25% of the gross income can only be used in a context that *both* a field for declaring the pre-paid taxes *and* a field for declaring gross income exist. Such a context may be a form area, a form or the whole electronic service. The same remark holds *in some cases* for instructions, legislation and documentation. For example, the instructions on how to fill in a VAT number and the legislation defining the form of VAT numbers should *always* be coupled with a field accepting input corresponding to a VAT number. However there do exist cases where instructions, legislation and documentation may be reusable (nearly) independently of other components, such as instructions regarding generic navigation issues, working with forms and fields or legislation related to the use of electronic services in general. Thus, two categories for instructions, legislation and documentation may be identified (a) items that *should be coupled* with other building blocks to achieve "near independence" and (b) items that are nearly independent on their own right. Packaging of the first category is discussed below, while items of the second category are allowed to exist autonomously in the reusable component repository.

In order to better serve the reusability purposes, the building blocks of electronic services are repackaged as follows:

A *form field* is bundled together with the related instructions, the validation checks and the legislation or any other documentation that pertains to the specific form field. Such a bundle is called a *transaction service element* (TSE). An examples of reusable component at TSE level is *the US SSN*, which is a nine-digit number with dashes after the third and fifth digit, bundled together with the validation checks enforcing such a form, instructions for the end-user on how to enter a valid SSN and any related legislation and documentation. Note that reusability scope of such a bundle is quite high, since it may be used in any electronic service involving US SSNs.

Figure 2 – Entering dates without typing

A *form area* is packaged along with the instructions, validation checks and legislation or any other documentation that are related to the form area as a whole. The form area package also contains the transaction service elements that appear within the form area and, transitively, all the instructions, the validation checks and the legislation bundled with the individual transaction service elements. The validation checks packaged into a form area may involve any transaction service elements included in the form area. Such a package is called *transaction service element group* (TSE group). A first example of a reusable TSE group may be a bundle of three individual TSEs allowing the user to enter dates without typing, as illustrated in Figure 2 (typing dates is a common source of errors in electronic services, thus the ability to enter dates without typing is strongly desired [6]). The TSE group contains the three individual TSEs, the related validation checks (e.g. disallowing the specification of the 31st of February), and all pertinent instructions and documentation. A second, more specialised example is that of a TSE group allowing the electronic service user to enter a taxpayer's country and VAT number. Since different rules apply to VAT number in different countries, the validity of a VAT number can only be determined if the country it has been issued in is known, thus the proper validation check cannot be

directly associated with any individual TSE. The instructions, documentation and legislation in this TSE group will also need to cover all countries that can be specified in the relevant TSE.

A *form* is bundled together with the instructions, validation checks and legislation or any other documentation that are related to the form as a whole. The bundle also contains the visual form layout, the individual transaction service elements and transaction service element groups that appear on the form and, transitively, all the instructions, the validation checks and the legislation packaged with the individual transaction service elements and transaction service element groups. Validation checks in the context of the form may reference any field appearing on the form either directly or indirectly through a transaction service element group. Such bundles are called *transaction service forms* (TS forms). An example of a reusable form is a form collecting personal details, which is directly reusable in any electronic service. In some cases, amendments may be needed, as for example in the detail forms of the Greek VIES acquisitions and VIES deliveries services (accessible through http://www.e-oikonomia.gr, for registered users only), between which only minor differences exist.

Finally, a whole service is packed along with the forms that constitute it, the instructions, validation checks and legislation or any other documentation that pertain to the service as a whole. Validation checks in the context of a service may reference any combination of fields appearing within the transaction service, either within a single form or on multiple forms. Such packages are called transaction services (TS).

The packaging described above tackles both the issues of triviality and independence, since any bundle is (a) all packages contain significant information and constitute an amount of work that is not easily repeatable in its full extent and (b) each package is self-contained and can be meaningfully used in an appropriate context. Note that all remaining component reusability criteria are also met:

(1) *Components are replaceable*, since any component may be replaced by any other component of the same class, in any valid context.

(2) *All components perform a clear function*. For instance, fields may be filled, validated and included in transaction service element groups and transaction forms while forms can be submitted, validated and included in transaction services. In both cases, also, the associated instructions and legislation may be viewed.

Notice that all the component reusability criteria are directly met for the *communication with back-office systems* building block, since it is definitely non-trivial, it is independent to a large extent of all other functions, some implementation may be easily replaced by another component of equivalent functionality and its function within the system is clear. As note before, electronic services are usually delivered through a dedicated system, which communicates the submitted data to an installed organisational IT system. This task can be further analysed into the following subtasks:

1. collection of the values submitted by the citizen
2. transmission of the collected values to the back-end system
3. restructuring of the information in a form appropriate for the back-end system and insertion into the relevant repository of the organisational workflow.

Out of these three subtasks, the two first may be standardised, since the collection of values from a specific service delivery environment is usually performed in a standard way (e.g. in a PHP [7] environment values are usually stored in the user session variable HTTP_SESSION_VARS; in ColdFusion [8] programmers usually store such values in an array-type variable; in a Java environment values are generally collected through the relevant bean [9]). The transmission between the two systems can also be performed using a number of standard techniques, e.g. RMI [10] or XML [11] messages on top of TCP/IP [12] or SSL/TLS [13]. The third subtask is highly dependent on the actual organisational information system, so no global solution can be provided; instead, the organisation's IT staff can write custom modules to perform this subtask, as detailed in [14].

3 PROMOTING REUSABILITY IN ELECTRONIC SERVICES

In order to promote reusability in the development of electronic services, developers must be empowered to (a) locate reusable components that are pertinent to the task at hand (b) customise these components to exactly suit the task and (c) create and make available to other users their own reusable components. To facilitate this task a *reusable component repository* is introduced, complemented with tools enabling users to browse, query, populate and customise its contents. The repository approach is illustrated in Figure 3.

The transaction element management (TSE management) facility enables users to create *templates of reusable TSEs*. A reusable TSE template contains exactly the same information as an individual transaction service element (i.e. label, input area, validation checks, instructions, documentation and legislation), but is not directly used in transaction services. Instead, users create *instances* of this template and customise it to suit the needs of particular circumstances, since a TSE need not appear identical in all its occurrences. For instance, a TSE representing a person's VAT number may appear in a tax return form as "Taxpayer's VAT number" in the area for personal details, as "Landlord's VAT number" in the section in which housing expenses are declared and as

"Employer's VAT number" in the incomes section. Besides the changes in labels, the validation checks associated with each occurrence may need to be customised (e.g. the Taxpayer's VAT number is always mandatory while the landlord's VAT number is mandatory only if housing expenses are declared; the employer's VAT number may need to be verified to correspond to an enterprise, rather than an individual). Once a TSE template has been instantiated and (possibly) customised, it can be used within a form of transactional service. Note that customisation still possible after the establishment of the link between the instantiated TSE and the transactional service.

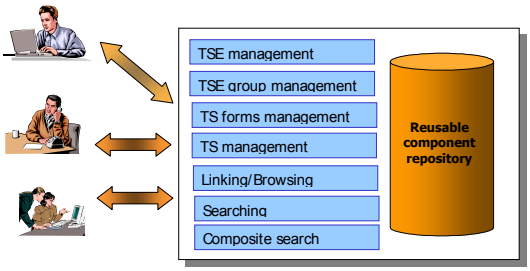

Figure 3 – Introducing the reusable component repository

A similar approach is used for TSE groups, i.e. users create instances of generic TSE groups, which then appropriately customise for use in services. For transaction service forms and transaction services, however, it was considered preferable to not introduce the concepts of transaction service form templates and transaction service templates, respectively, since the cases in which whole forms or whole transaction services will be reused are less frequent than the cases in which TSEs or TSE groups will be. Instead, for transaction service forms and transaction services a *clone* facility has been provided, which create exact duplicates of the source object. The developer can then customise any component of the cloned object.

For the reusable component repository to be effective, however, developers must be provided with appropriate tools to locate reusable components. The classic mechanisms for locating items within a repository are *searching* and *browsing*, which are both provided in the proposed approach. Through searching, users enter *patterns,* which are matched against the contents of the repository, and the components that qualify with respect to the matching are included in the result. The search pattern may include free text search, either in all sections of elements or in specific ones (e.g. label, documentation, validation rules, author, keywords or any combination of fields) and the type of the desired result may also be indicated (for example, "TSEs only"). The simple form of querying, however, can prove ineffective in the context of the reusable component repository, because search mechanisms usually target *single objects,* while the elements of the repository are *composite*, thus the relevant information is dispersed among several objects. Consider for example the case of a developer searching for a TSE group representing a citizen's details, i.e. name, surname, address, social security number and phone. Searching for an item containing the strings "Name", "Surname", "Address", "SSN" and "Phone" will probably fail, because each of these strings is contained within the relevant TSE, while the TSE group contains only links to these TSEs. Searching for a TSE group named "Citizen details" might also fail, because the TSE group may have been named "Citizen data", "Personal details" etc. In order to address this shortcoming, *structured searching* is provided, which allows for the developer to specify criteria that *contained elements* should fulfill, for the containing object to qualify for the result. Using structured search, the query to locate the citizen's details TSE group could be formulated as *retrieve TSE groups having (a) a contained TSE matching "Name" (b) a contained TSE matching "Surname"* and so on. Users are assisted in entering structured queries by a graphical user interface, illustrated in Figure 4. In the depicted query, the *type* of the result objects is specified (TSE group), it is also stated that the resulting object should be linked to (a) an object whose description matches "Name" and (b) an object whose description matches "Surname".

In structured queries, a *criteria relaxation* approach is followed, according to which an object may appear in the query result, even though not all defined criteria for linked objects are met. This feature has been considered useful since developers may thus locate reusable components that are *similar* to the components they seek, and can then instantiate (or clone) customise these components. In the previous example, a developer might locate a TSE group containing the TSEs "Name", "Surname", "Address" and "SSN", but not the "Phone", which could however be added to the specific instance created for the service under development. The relaxation degree (i.e. the number of criteria for linked objects that an item in the result set may not meet) can be set by the user stating the structured query.

Complementary to searching, browsing mechanisms are offered to the users. In this case, users are presented with a *classification scheme* (or *taxonomy*) for reusable components, and are able to drill down this scheme to locate the desired components (see Figure 5). The classification scheme may reflect the organisation's structure (e.g. by department), be based on the components' semantic aspects (e.g. income tax components, real estate components), or follow any other convenient structure. Multiple classifications may also be present, to provide for alternative concept paths for locating specific reusable components. The actual reusable components are located at the leaf nodes of the classification scheme, whereas non-leaf nodes correspond to classes of reusable components.

Regarding the structure of the classification scheme, it must be noted that although it is displayed as a tree,

the internal structure is a *direct acyclic graph*. This allows for linking the same reusable component into multiple concept categories, enabling developers to more easily locate an item. For example, the "Personal details TSE group" may be linked under the categories "Generic reusable components", "Income tax/generic reusable components", "VAT/ generic reusable components" and "VAT/periodic declaration" (slashes indicate drill down points within categories). The gain offered by multiple linking to concept categories incurs however a cost, since these links have to be established manually by domain experts. It must be noted though that linking may be performed incrementally after an item is placed in the repository, so no heavy burden is placed on the creator of an item to establish all the necessary links upon the insertion of a reusable component into the repository.

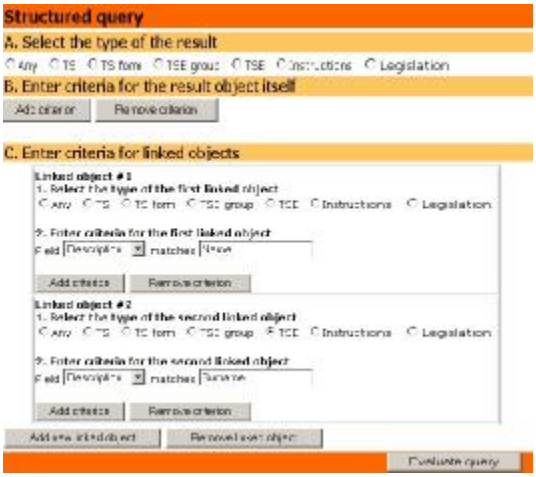

Figure 4 – Using structured search

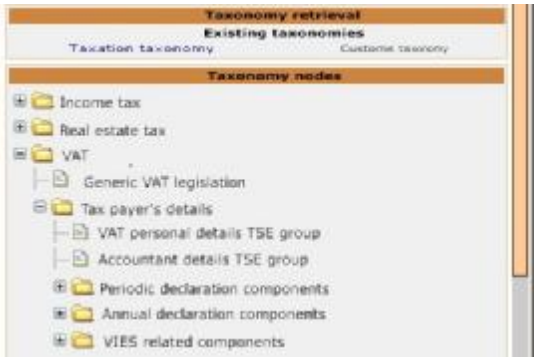

Figure 5 - Locating reusable components through taxonomies

4 CONCLUSIONS – FUTURE WORK

In this paper we have discussed the issue of reusability in the development of electronic services. We have showed that by appropriately decomposing electronic services in their constituent parts, it is possible to identify portions that may be reused across electronic services, minimising both the recourses needed for development and testing and the service roll-out time. We have also presented a repository-based development approach for electronic service, which allows for components to be placed within a repository and then be extracted for use in other services. Future work will focus on intelligent agents that will analyse the components created by users and automatically propose the use of existing components. Using semantics-based techniques, as those discussed in [15] for facilitating browsing and searching within reusable component repositories will be also investigated.

5 REFERENCES

[1] eEurope, 2000. Common list of basic public services. Available at http://europa.eu.int/information_society/eeurope/2002/ action_plan/pdf/basicpublicservices.pdf

[2] B. A. Burton, R. W. Aragon, S. A. Bailey, K. D. Koehler, L. A. Mayes, "The reusable software library", in Software reuse: emerging technology, IEEE Computer Society Press, 1988, pp. 129-137

[3] M. W. Price, S. Demurjian, "Analyzing and Measuring Reusability in Object-Oriented Designs", Proceedings of the OOPSLA conference, Atlanta, Georgia, 1997, pp. 22-33

[4] M. Fowler, "Analysis Patterns: Reusable Object Models", Addison-Wesley Professional, 1996.

[5] Philippe Kruchten, "Modeling Component Systems with the Unified Modeling Language", Proceedings of the 1998 International Workshop on Component-Based Software Engineering, 1998

[6] SmartGov consortium, "D41: User Requirements, Services and Platform Specifications", available from http://www.smartgov-project.org

[7] Lerdorf, R., Tatroe, K. 2002. Programming PHP. O'Reilly & Associates, 2002, ISBN: 1565926102

[8] Hewitt, E. 2001. Core ColdFusion 5. Prentice Hall, 2001, ISBN: 0130660612

[9] Englander, R., *Developing Java Beans*, O' Reilly, 1997

[10] Sun Microsystems (1999) JavaTM Remote Method Invocation, available at java.sun.com/j2se/1.3/docs/guide/rmi/index.html

[11] World Wide Web Consortium (2001) The XML Specification, available at http://www.w3.org/xml

[12] Stevens W. R., TCP/IP Illustrated: Volume 1. The protocols, 10th Edition, Addison-Wesley Pub Company, 1997

[13] Thomas S. A. (2000) SSL & TLS Essentials: Securing the Web, John Wiley & Sons, ISBN: 0471383546

[14] C. Vassilakis, G. Lepouras, S. Rouvas and P. Georgiadis, "Integrating e-Government Public Transactional Services in the Public Authority Workflow", Electronic Government J., vol. 1, Inderscience Publications, pp. 49-60.

[15] Haining Yao, Letha Etzkorn, "Towards A Semantic-based Approach for Software Reusable Component Classification and Retrieval", Proceedings of the 2004 ACM SouthEast Conference, April 2-3, 2004, Huntsville, Alabama, USA, pp. 110-115.