Reusability in Governmental Electronic Services

George Lepouras, Costas Vassilakis
Dept. of Computer Science and Technology,
University of Peloponnese

INTODUCTION

Reusability in the everyday life is the capacity of using existing objects or even concepts again in the same or other contexts. When applied to information systems reusability is the capability of using the same parts of an application in other applications or in other contexts. As defined by IEEE (IEEE, 1990) *reusability* is the degree to which a software module or other work product can be used in more than one computing program or software system. Although Rapid Application Development environments such as MS Visual .NET™ (Visual .NET, 2005) and Delphi™ (Borland Delphi, 2005) have to some extent employed reusability of components to aid the fast implementation of software applications, the extent to which existing objects can be used again in the implementation of new software systems is usually limited to basic building blocks of the interface. One problem that hinders reusability of larger building blocks is the fact that once a component which encompasses a number of objects is built it also encapsulates algorithms (sets of well-defined instructions that perform a task) in the form of code that define the functioning of the component. Tight coupling between the program logic and the program code makes portability of the component between applications difficult, when even small modifications in the program logic are required. A second problem that hinders reusability is that even when the same

component can be directly used between applications, recognizing that such a component exists and retrieving it, is not an easy task.

Reusability plays an important role in software development industry. If a set of well-defined components is available, valuable resources can be saved by utilizing again these components. As Rich Seeley observes (Seely, 2003) “as part of that cost-saving message, Gartner recommends vendors and consultants focus on reusability of Web services applications and components”.

B A C K G R O U N D

Electronic government is an area where a lot of development effort is lately devoted to. Electronic government aims to promote the use of electronic means, mainly electronic services to facilitate communication and interaction between civilians or businesses and the government. According to the European Commission (European Commission, 2000) “*transaction services, such as electronic forms, are perceived as the future of electronic government*”.

An electronic transaction service is usually the electronic counterpart of an existing service, implementing the business process logic involving the filling and submission of forms containing the necessary data, the processing of these data according to rules derived from laws and regulations and finally the return of a reply to the user.

In order to implement transactional electronic services the collaboration between a number of experts is required. To this end, reusability can help by minimizing the effort needed for developing online transactional services. Electronic government offers a prominent area for the application of reusability since services offered to citizens from the same or different public authorities have common parts that could be reused between their electronic counterparts. However,

in order to have effective reuse of components the two main problems previously described have to be solved. Back in 1995 Dusik and Katwijk (Dusik, 1995) identified the importance of a software development environment in which reuse, in various forms, would be an integrated element. As Gall et al. (Gall, 1995) noted the goal for reusability should be to create a software development process based on the "use" rather than the "reuse" of standard components. The approach used during the SmartGov project (SmartGov, 2001) involved the design and implementation of a e-service development environment that would enable developers and domain experts to use components that they or other users had created to create their own transaction services.

SMARTGOV APPROACH

In contrast to simple information services, transaction services allow users to submit their data and in response the Public Administration performs a service such as the issuing of a certificate or the tax clearance. Transaction services allow the user to perform common services online, implementing thus one of the main objectives of the electronic government, namely the facilitation of the interaction between civilians and businesses with the public authorities.

To be able to implement reusability effectively one has to start by decomposing a transaction service to its basic building elements. In the first level an electronic service consists of a number of forms the user is required to fill in. In the case of short documents one form may be enough, where for lengthy documents more than one forms may be necessary. A form itself may comprise of several *areas*, and each area commonly contains individual

fields, which are conceptually interrelated. The term *field* denotes the equivalent of a paper form field, which in the electronic service may be implemented as text input field, selection list, radio button group, etc.

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. As Shaw pointed out (Shaw, 1995) 90% of most applications code goes into system or administrative code, like user interface code and back-end processing. Thus reusability of objects combining the visual part of the field and the inherent processing logic is crucial. Objects greatly increase software reusability and simplify the software development process (Fan, 2000).

As noted earlier for a reusability approach to be effective two issues have to be tackled: the tight coupling between the logic and the program code (i.e. between what we aim to achieve and the code that implements it) and the implementation of suitable mechanisms for retrieving components. The first issue can be solved by providing facilities to customize components without the need for completely rewriting the program code while the second can be

solved by offering mechanisms for locating components pertinent to the tasks at hand and mechanisms for publicizing components to other user.

To facilitate these tasks a *reusable component repository* is introduced, complemented with tools enabling users to browse, query, populate and customize its contents. The repository approach is illustrated in *Figure 1*.

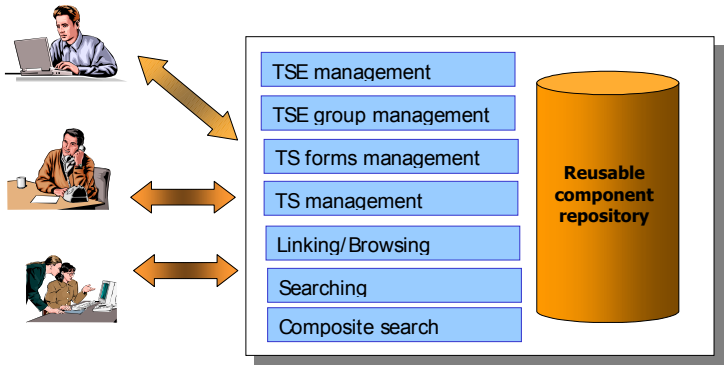

Figure 1 – Introducing the reusable component repository

In the proposed approach the idea of a basic building block is introduced. A *Transaction Service Element* (*TSE*) is the equivalent of paper based form field. However, in contrast to a simple field, the TSE has more into it. A TSE can have a multilingual label, the field for inputting data, validation checks for checking the conformity of data to rules, instructions, documentation or even legislation that applies on the field. The component repository holds *templates of TSEs* and *of groups of TSEs*. 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, but is not directly used in transaction services. Instead, users create *instances* of this template and customize 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 customized (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) customized, it can be used within a form of a transactional service. Note that customization is still possible after the establishment of the link between the instantiated TSE and the transactional service. A similar approach is used for TSE groups, i.e. users create instances of generic TSE groups, which can then appropriately customize for use in services.

In the approach adopted by SmartGov it was not considered appropriate to 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 creates exact duplicates of the source object. The developer can then customize any component of the cloned object.

So far, the notion of the reusable component repository has been described. However, the presence of a repository containing customizable objects does not automatically guarantee the effectiveness of the reusability approach. It has to be complemented with tools that will allow efficient management of the components. As already stated, for such a repository to be useful it is of the essence to provide efficient navigation and searching facilities that will allow users to locate the elements they want to view or modify. Moreover, when new services are created or existing services are modified, it is very desirable to be able to *reuse* existing components implement the needed functionality. For instance, most services have a special form or form area in which the personal details of the service user are displayed; when creating a new service, it is beneficial to re-use this form from en existing service, since development resources and time are saved, testing has already been done and uniformity across services is achieved.

Braga et al. (Braga, 2001) have proposed the use of an *ontology* to aid retrieval of components that exist in distributed repositories. As defined in (Noy, 2001), an ontology is a formal explicit description of concepts in a domain of discourse (called classes or entities), properties of each concept describing various features and attributes of the concept (called slots, roles or properties), and restrictions on slots (called facets or role descriptions).

In the SmartGov approach a simpler yet efficient navigational scheme was used based on *taxonomies*. According to WhatIs?Com Online Encyclopaedia (WhatIs, 2005) a *taxonomy* is a classification according to a

pre-determined system, with the resulting catalog used to provide a conceptual framework for discussion, analysis, or information retrieval. The basic difference between an ontology and a taxonomy is that an ontology defines not only the concepts (the classes) but also their properties, as well as possible restrictions on how the classes and properties can be instantiated. In a taxonomy, concepts are classified hierarchically, with each concept being a separate node in the hierarchy. Nodes appearing in the lower level of the hierarchy are known as leaf nodes. Under this scheme, an organisational taxonomy is built with broad categories at the first level, which are refined into smaller categories at the second level and so on, until the desired level of detail (usually 5-7 levels) (Fraser, 2003). Platform users that create elements can link them at any category node, either leaf or non-leaf; linkage of elements with nodes can also be modified at a later stage. Users needing to locate an element, start from the top node of the taxonomy and drill down the categories (Figure 2). Once an element is reached and displayed, the navigational facilities should allow the platform user to move to any other element linked to the current one; for example, if a field is displayed, the user should be able to view the form(s) that this field appears on, the groups it participates in, the validation checks it is involved in, examples illustrating its usage, legislation pertaining to it and so on. The information enabling the platform to display these links has been already entered by the relevant stakeholders, either as an indispensable part of the element definition (e.g. when defining a form the user selects the fields that should appear on it; the

definition of a validation check references the involved fields and so forth), or as express linkage (e.g. linkage of legislation and documentation to elements). Linkage may also be implicit and derived by the context of actions – e.g. if the user selects the "Create an example" action when editing a form, the example will be linked with the form being edited.

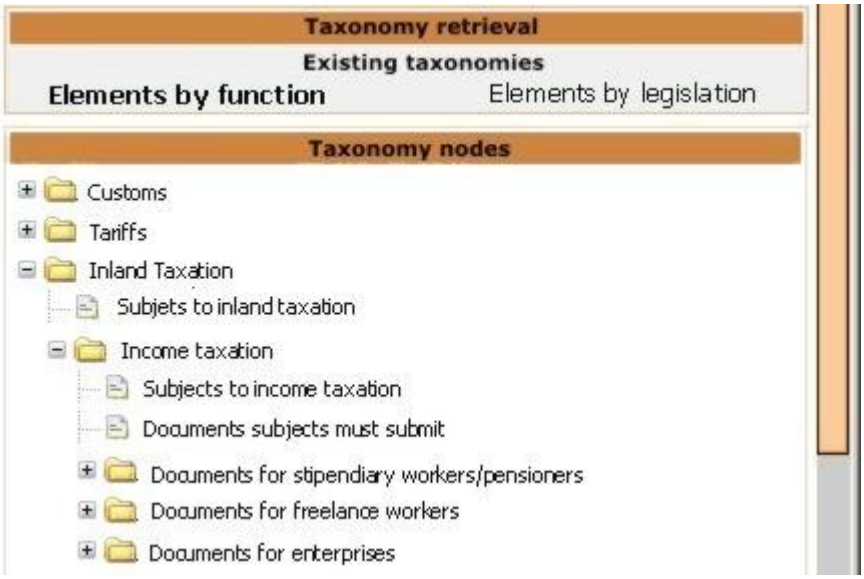

Figure 2 – Taxonomies for navigation in the repository

Finally, the organisation may want to define multiple taxonomies, as is the case in Figure 2. With multiple taxonomies, different classification schemes can be supported to facilitate the work of users with diverse expertise or interests. In the example of Figure 2, two taxonomies are used: the "elements by function" taxonomy is addressed to domain experts specialising on different taxation items, and the "elements by legislation" taxonomy, addressed to legal advisor. While support of different taxonomies is beneficial for users that try to locate elements, it places an extra burden for

element authors, since a link must be established for each distinct taxonomy. Semi-automatic classification schemes may alleviate this problem.

The search mechanism allows users to 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. labels, descriptions, author, keywords, content [for document-type elements only, i.e. examples, documentation and legislation] or any combination); users may also designate and the type of the desired result (e.g. *fields only*, or *examples and legislation*).

One issue that must be addressed with searching within the repository is that standard search engines examine *individual objects* whereas when searching the repository the information stated in the search pattern may be dispersed across several repository elements, perceived however by the querying user as a single entity. For instance, if the user enters a query requesting objects containing *all the words* "Name", "Surname", "Address" and "Id number", there may exist no single object containing all these words and, consequently, a standard search engine would produce no results. However, users would *expect* a field group "Personal details" to be retrieved by the query, because its elements *collectively* satisfy the search criteria. In order to tackle this issue, a modified search engine should be used for searching the repository. The modified search engine flags that an element matches a pattern if either the element itself or any of its *contained elements*

matches the pattern. The “containment” relationship is defined as follows: *transactional services* contain *forms*; *forms* contain *form element groups* and/or *individual form elements*; and *form element groups* contain *form elements*. The containment relationship is also transitive, e.g. if a form element is contained in a form, it is transitively contained in any transactional service containing the specific form. Finally, validation rules, examples and documentation are directly “contained” in any element they are linked to. Another complementary facility that can be used during searching is a dictionary of synonyms. If for example, the user searches for “Surname” and an object with description “Last Name” exists, the dictionary can help in retrieving this object.

FUTURE TRENDS

Although the use of taxonomies can help in managing and reusing code, an ontology can offer a richer and more complete image of the organization which produces the service. To this end, know-how acquired from the SmartGov project can be used to incorporate ontologies as a mechanism for semantically managing reusable components. Furthermore, since the organization can change in time, a versioning system for the ontology can be introduced which will allow finding the elements that become obsolete as well as the history of the changes.

CONCLUSION

It is widely recognized today that reuse reduces the costs of software development (Mili, 1995). However, in order to efficiently implement reusability, a system is required that will enable the management of code fragments according to their logic. In the framework of the SmartGov project a knowledge based platform was implemented that allows semantic classification of transactional service elements, fast and easy copying and modification of existing code and management of the service logic by means of taxonomies. The proposed approach has been proven (SmartGov Consortium, 2004) to offer a viable and efficient solution to implementing transaction services by means of reusable components.

R E F E R E N C E S

Borland Delphi Home Page (2005). Retrieved November 17, 2005, from http://www.borland.com/delphi/

Braga, M. M. R., Mattoso, M., Werner, M. L. C. (2001). The Use of Mediation and Ontology Technologies for Software Component Information Retrieval. *Proceedings of ACM SSR'01*, Toronto, Ontario, Canada, 19-20

Dusink, L., Katwijk, J. (1995). Reuse Dimensions. *Proceedings of ACM SSR '95*, Seattle, WA, USA, 137-149

European Commission (2000). Public Sector Information: A Key Resource for Europe, *Green paper on Public Sector Information in the Information Society*, Retrieved November 17, 2005, from http://www.cordis.lu/econtent/publicsector/greenpaper.html

Fan, M., Stallaert, J., Whinston, A.B. (2000). The adoption and design methodologies of component-based enterprise systems, *European Journal of Information Systems*, 9(1), 25-35

Fraser, J., Adams, N., Macintosh, A., McKay-Hubbard, A. (2003). Knowledge Management Applied to e-Government Services. *Proceedings of the KMGov2003 Workshop*, Rhodes, Greece.

Gall, H., Jazayeri, M., Klosch, R. (1995). Research Directions in Software Reuse: Where to go from here? *Proceedings of ACM SSR '95*, Seattle, WA, USA, 225-228

Institute of Electrical and Electronics Engineers (1990). *IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries*. New York, NY

Mili, H., Mili, F., Mili, A. (1995). Reusing Software: Issues and Research Directions, *IEEE Transactions on Software Engineering*, 21(6), 528-562

Noy, F. N., McGuiness, L. D. (2001). Ontology Development 101: A Guide to Creating Your First Ontology, *Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880*

Seely, R. (2003). Gartner: Web services projects roll along, *AdMag.com*, Retrieved November 17, 2005, from http://www.adtmag.com/article.asp?id=8076

Shaw, M. (1995). Architectural Issues in Software Reuse: It's Not Just the Functionality, It's the Packaging, *Proceedings of ACM SSR '95*, Seattle, WA, USA, 3-6

SmartGov Consortium (2004). *SmartGov Project D91 - Evaluation of project results*. Retrieved November 17, 2005, from http://www.smartgov-project.org/ index.php? category = results

SmartGov (2001). IST-2001-35399, A Governmental Knowledge-based Platform for Public Sector Online Services, Retrieved November 17, 2005, from Project Website: http://www.smartgov-project.org/

Visual Studio Home Page (2005). Retrieved November 17, 2005, from http://msdn.microsoft.com/vstudio

WhatIs (2005) IT Encyclopedia and Learning Center. Retrieved November 17, 2005, from Home Page http://whatis.techtarget.com/

T E R M S A N D D E F I N I T I O N S

Clone (function): a function which copies all aspects of an existing object including visual appearance, parameters and links to code affecting its behavior, creating thus an identical copy of the original object.

Field: The term field denotes the equivalent of a paper form field. Although in a paper based form a field is usually a box that the user has to fill in, in an electronic service the same field may be implemented as text input field, selection list, radio button group, etc.

Ontology: An ontology is a set of concepts for a certain domain, connected together with inheritance relationships and each of them having a set of attributes.

Pattern (search): a string containing alphanumeric and possibly special characters (such as wildcards) used as a target to search for. In the simplest case the string contains a word (or part of it), while in other cases it can contain multiple words or regular expressions.

Reusability: The extent to which a software module of an existing application can be used in other applications and/or in other contexts.

Reusable component repository: A repository that can hold reusable components. To be usable the repository is complemented with tools that allow the managing of components (i.e. the storing, categorizing, retrieval and dissemination of components).

Software module: a software component that performs a well defined function and is independent of other components.

Taxonomy: A hierarchical classification of concepts for a certain domain. The main difference between a taxonomy and an ontology is that the taxonomy lacks the set of attributes for each concept. In a taxonomy concepts are classified hierarchically, with each concept being a separate node in the hierarchy. Nodes appearing in the lower level of the hierarchy are known as leaf nodes.

Transaction Service Element (TSE): *Transaction Service Element* (*TSE*) is the equivalent of paper based form field. However, in contrast to a simple field the TSE has more into it. A TSE can have a multilingual label, the field for inputting data, validation checks for checking the conformity of data to rules, instructions, documentation or even legislation that applies on the field.

TSE template: A TSE template is the equivalent to the a class definition in object oriented programming. It can be instantiated to a TSE or modified to create a new TSE template.