Tailorability in the Context of E-Government Information Systems: An Approach

George Lepouras
Department of Computer Science and Technology, University of Peloponnese, Terma Karaiskaki, 22100 Tripolis, Greece,
Tel: +302710372201, Fax: 2710 372160, G.Lepouras@uop.g

Anya Sotiropoulou
Department of Computer Science and Technology, University of Peloponnese, Terma Karaiskaki, 22100 Tripolis, Greece,
Tel: +302710372201, Fax: 2710 372160, anya@uop.g

Dimitrios Theotokis
Department of Computer Science and Technology, University of Peloponnese, Terma Karaiskaki, 22100 Tripolis, Greece,
Tel: +302710372201, Fax: 2710 372160, dtheo@uop.g

Costas Vassilakis
Department of Computer Science and Technology, University of Peloponnese, Terma Karaiskaki, 22100 Tripolis, Greece,
Tel: +302710372201, Fax: 2710 372160, costas@uop.gr

ABSTRACT

The ever changing environment information systems model, and in particular e-government ones, intensifies the need for systems that are able to easily, efficiently and transparently adapt to changing environments. Accommodating unanticipated changes implies that systems must be able to adapt to changes occurring in and evolve in step with their changing environment. Adaptation is concerned with monitoring, analysing and understanding the patterns of the user's interaction with the system. Similarly, an information system is said to be evolutionary if it can be purposefully used in a dynamic environment. E-government information systems, in virtue of their nature and function, are driven by the need to adapt and evolve. This suggests that the design and implementation of such systems must provide the necessary infrastructure for evolution and adaptability. In other words, e-government information systems must abide to the Tailorable Information Systems paradigm. In this work we present a case study for the development of a Tailorable e-government information system.

INTRODUCTION

The need for information systems' adaptability and evolution has been well established by researchers and practitioners alike [Kiczales et al., 1997, Lycett and Paul, 1998a, Lycett and Paul, 1998b, Patel and Paul, 1998, Patel, Gardner and Paul, 1995, Stamoulis, Kanellis and Martakos, 2001, Stamoulis, Martakos and Introna, 1998]. It is argued that information systems are not flexible in terms of accommodating changes, both anticipated and unanticipated ones, due to the traditional systems design and implementation approaches, because systems are built to an exact, predetermined specification [Stamoulis et al., 2003]. Fixing design decisions during systems development leads to code scattering and tangling when introducing new functionality as this emerges from new requirements [Stamoulis et al. 2003]. This, in turn, leads to poor traceability between requirements, designs and code, a fact that renders information systems hard to maintain and evolve [Harrison and Ossher, 1993]. Thus it becomes evident that systems exposed to changes, and in particular unanticipated ones, should, in order to remain flexible in terms of accommodating such changes, adopt a deferred design decisions paradigm [Stamoulis et al., 2003], whereby a system is built on the principle that it will evolve and, because of this, it should provide the necessary infrastructure and architecture for transparently encompassing such changes. Although a number of approaches have been proposed for building information systems able to accommodate unanticipated changes, such as Aspect-oriented programming [Kiczales et al., 1997], Design Patterns [Gamma et al, 1994], Service-oriented composition [Loverdos et al., 2002], we have adopted the use of the ATOMA framework and runtime system [Theotokis et al, 1997] which supports evolution by enforcing behavioural separation of concerns [Theotokis, 2003]. According to the behavioural separation of concerns approach a system consists of its basic behavioural landscape elicited during the analysis and requirements phase, and the behaviour that is, or rather will be, acquired during a system's lifetime. These are modelled separately and the later is incorporated into the original system by means of techniques that permit behavioural evolution. The focus of this work is on the design and implementation of an e-government system that shares a registry between various public administration authorities (PAs). The key notion behind this system is that the registry realises specific to its maintenance behaviour, while each civil service organisation provides the organisation-specific behaviour which is incorporated, dynamically, to the registry's functionality and may be shared amongst other civil service organisations.

The rest of this paper is organised as follows: Section 2 provides a detailed description of the application emphasising the need for a solution that supports transparent behavioural evolution and a concise description of the ATOMA framework and runtime system. Section 3 presents the use of the ATOMA framework and runtime system as the implementation platform for the application is question focusing on the benefits that emerge, as well as the realisation of a tailorable deferred design decisions based information system. Section 4 investigates related to the proposed approaches for behavioural evolution and provides insights as to why the proposed approach is considered to be more appropriated for the task in hand. Finally, section 5 concludes the paper.

BACKGROUND

Nowadays, most public administration authorities (PAs) and organisations have their own information systems, which support the organisational business processes; within these business processes some real-world entities interact with one another (e.g. a citizen buys a car) or with the public administration (e.g. a citizen submits a social security benefit claim). Within the context of public administration, as a whole, the same real-world entities participate in multiple business processes occurring in different public authorities; it would be thus anticipated that each real-world entity is represented using a single object that is shared by all public authorities that are authorised to access it. This object would consolidate all the information needed by all public authorities that include the respective real-world entity within their business processes, and would be equipped with all the necessary behavioural elements that would allow the public authority information systems to operate on it. Each such object would be stored only once, guaranteeing thus absence of inconsistencies and minimising storage requirements. Physical storage of objects representing some real-world entity would be undertaken by a specific PA information system, which will provide creation, retrieval and update facilities to other PA information systems; the selection of the PA system that will physically store a specific object type may be based on technical information (e.g. an object type is stored into the information system it is mostly accessed from), administrative information (e.g. an object type is stored into the information system of the PA that is *de jure* responsible for the corresponding real-world entities etc).

In most cases, however, each PA information system that manipulates some real-world entities stores its own copy of objects representing these entities, despite the consequent increase in development and population time, and the risk of inconsistencies. The reason for this practise is not the inability to share information between different information systems, since technology provides ample means for implementing information sharing between information systems (federated databases [Chorafas and Steinmann, 1993], web services [Linthicum, 2003], XML format [Benz and Durant, 2003] etc). This practise may be attributed to the following causes:

1. Different public administration authorities may require differently organised views of the same data. For example, in the ministry of Finance subjects are mainly identified through their VAT number, while in the Department of Health subjects are mainly identified by their social security number (which is the case in most European Union countries).
2. Given that all PA information systems access the same objects, their views regarding the object schemata should be consistent. This hinders the maintenance, since the schema modifications to an object type due to requirements posed by some PA will necessitate maintenance activities to be performed on all PA information systems accessing the same object type. The existing objects will also need to be adapted to the new schema in a global fashion, which is not a trivial task.

Between these two extremes, a more viable solution would provide for the ability to differentiate between the aspects of a real-world entity that are of global interest, and the aspects that are of interest only to a specific information system. The globally interesting aspects would then be stored in a centralised repository, from where they are *imported* into each information system that needs to manipulate this entity. The import procedure in such a case, however, is not limited to the mere establishment of a link towards the object representation in the centralised repository: firstly, the imported object should be *coupled* with the aspects that are of interest to the local information system only (and are stored into it). Secondly, *extended behaviour* should be assigned to the object in order to (a) provide the necessary methods implementing the business processes of the PA (which may include both global and local aspects) and (b) override default behaviours of the global object, if needed, tailoring thus the object to the local needs and facilitating the creation of a "localised view", which may be required in the context of the business processes.

The following figure illustrates an example of the proposed approach. The information system maintained by the Ministry of Interior stores a global registry for a country's citizens. This information system *exports* the *Person* entity (corresponding to the citizen), which is then *imported* by the information systems of the Ministry of

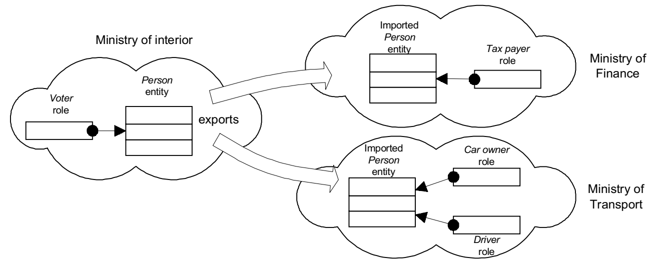

Figure 1 – Entity import/export and role assignment in the proposed approach

Finance and the Ministry of Transport. Each of the latter two information systems associates to the imported entity the roles (i.e. data and behaviour) that are of interest to its function (the Ministry of Finance assigns the *tax payer* role, while the Ministry of Transport assigns the *car owner* role and the *driver* role). Note that, the information system hosting the entity may assign to it roles pertaining to the information system's function (in this example, the Ministry of Interior assigns the *Voter* role to the *Person* entity).

THE ATOMA FRAMEWORK

The ATOMA framework emerged from the need to provide a mechanism for incorporating, dynamically, behaviour into existing information systems. In doing so, a system is modelled and realised in terms of two key notions: basic entities and roles. The former models the fundamental behavioural blocks of a system providing only the necessary behavioural landscape of the most abstract version of the entity modelled. For instance the entity *employee* will be modelled as the entity *person* without an employee related behaviour. Roles are used for modelling functional behaviour related to a basic entity. In this case employee will be a role. Similarly, doctor or taxpayer will also be roles. Basic entities are behaviourally modified by roles. Hence, the role employee will be "added" to the basic entity person when a particular person becomes an employee and will be removed from it when the person in question ceases to be an employee. Furthermore, role acquisition and removal can take place during runtime.

The framework is based on the object-oriented paradigm providing the necessary enhancements for the dynamic modification of objects' behavioural landscape. As such, basic entities are modelled (realised) as classes1, while acquirable behaviour as roles. According to Sowa, entities relate to the essence of the concept of notion they represent, while roles depend on an accidental relationship to some other entity [Sowa 1984]. Roles are themselves first class objects, but unlike classes cannot exist on their own. They have to be assigned to a class or class instance, in order to activate the functionality they provide. Classes and roles form the specification layer of the framework. Instances of classes and roles form the provision layer of the framework. These two layers are in accordance with the principles of the object-oriented paradigm. Responsible for the modification of an object's behavioural landscape is the composition layer. Specifications describing the composing and decomposing of classes and roles and their instances are registered with the composition layer, which is responsible for performing the compositions or decompositions. Activation of such specifications can occur either statically or dynamically. The former concerns the compile time phase of the system whereas the later is associated with the runtime phase of a system. Roles can be composed with classes or their instances. For example an instance of the *person* class may become a student. Roles can also behaviourally modify other roles. For instance, the *manager* role may be composed to the *employee* role, thus providing the *manager* role to an instance of the class *person* through its *employee* role. The key issue to note here is that through this approach there are no statically bound relationships between classes nor between instances and such relationships are established and abolished according to emerging requirements at runtime. A more detailed explanation of the ATOMA framework is beyond the scope of this paper. The interested author for the theoretical background of the ATOMA framework is referred to [Theotokis et al., 1997, Theotokis, 2001]

TAILORING E-GOVERNMENT INFORMATION SYSTEMS

E-government systems, as mentioned before, due to their nature should provide means for tailorability. In this section we present an example case where behavioural evolution as supported by the ATOMA framework enables a registry of citizens' data to be shared amongst different PAs without having to modify the registry's schema or common behaviour to meet each PA's needs.

Assume that the registry office, one of participating PAs, is responsible for registering all new born citizens, updating their information during their lifetime and finally removing or archiving all information related to a deceased citizen. Further, suppose that the Department of Transport uses the same registry when a citizen obtains a driving license or when a citizen buys a car. Finally, for the example's sake, the Department of Finance uses the registry in question for taxation purposes. A key issue to note here is that each of the aforementioned PAs uses a different identification scheme for the citizens in the registry. The Registry office identifies citizens by their social security number issued at birth, the Department of Transport by their driving licence id and finally the Department of Finance by their VAT number. It also must be noted that each PA has its own information system to perform its function and a localised registry in which PA specific information is kept. One of the issues arising from such a scenario is that in some cases the function of one PA relies on information obtained from the information system of another PA. Furthermore, according to emerging needs a PA may need to modify or enhance its information system and as such other PAs that utilise its information systems must always have the updated version of the information system's behavioural landscape.

Let us now consider the structure of the registry and the function of the PAs information systems. In order to keep the example simple we only provide the necessary for the example information instead of the whole picture. The common citizen registry contains amongst other information the name, surname, middle name, father's name, date of birth, social security number, and address of the citizen. The Department of Transport makes use of the information system supporting the Department of Finance in order to issue the yearly road permit for a citizen's car. Finally, the Department of Finance requires the functionality of the information system used in the Department of Transport in order to calculate the tax corresponding to citizen's car.

Given that shareable entity between all three information systems is the entity *person* as depicted by the registry's schema, when a citizen becomes a tax-payer an instance of the *taxpayer* role is added to the person in question, by the Department of Finance information system, and is kept to that department's localised registry. Similarly, when a citizen obtains a driving licence an instance of *drivingLicenceHolder* role is created and kept by the Department of Transport. The same holds when a citizen becomes a car owner. An interesting element of the ATOMA framework and runtime system is that for each car a citizen owns a different role instance will be created. Each role provides all the behavioural and structural characteristics associated with the aspect it models. For example, the role *carOwner* encapsulates information about the model and make of the car, the engine capacity based on which the car tax is calculated etc. It become evident that when new behavioural requirements occur they are modelled and realised as roles and "added" to existing information systems so that the underlying entities manipulated by them are behaviourally enhanced.

However, the benefit of the role based behavioural evolution is not limited to the localised information system. Let us consider the scenario where the tax due for a taxpayer should be determined. In this case the information system of the Department of Finance requires all tax related information that the information systems of other PAs can provide. In our specific example a request to the information system of the Department of Transport to export all roles associated with a given citizen will be issued. The roles will be exported and assigned to the localised representation of the citizen in question within the information system of the Department of Finance. Due to the nature of roles, provided that their public interface is kept consistent, any behavioural changes they undertake will be transparent to the other information systems. This offers a great degree of flexibility since evolution in an information system does not prohibit the function of another.

Finally to conclude the example, consider the case when a citizen is deceased. In this case the Registry Office will update the information in the central registry and will notify all other information systems which in turn they will remove all roles attached to the their localised representation.

RELATED WORK

A number of alternative approaches to behavioural evolution and tailorability have been proposed over the years, including but not limited to Design Patterns, Subject-Oriented Programming, Aspect-Oriented Programming and more recently Service-Oriented Composition. In this section we present the key ideas underlying these approaches focusing at the same time on the advantages that our approach provides.

Design Patterns

Design Patterns as proposed by Gamma et al [Gamma et al, 1994], inspired by the work of Alexander [Alexander, 1979] aim to provide descriptions of "best practice" designs for code reuse. In particular behavioural design patterns address the issue of behavioural evolution, as their name implies, in a manner that objects can modify their behavioural landscape in a given context by adopting behavioural adjustments from a predefined set applicable to the given context. Design Patterns rely on the concepts behind aggregation / parameterisation plus inheritance. Consequently the behavioural modifiability portraying Design Patterns exhibits a severe limitation: design decisions are fixed when realizing the pattern a fact that reduces the scope of behavioural evolution. In fact behavioural design patterns fail short of accommodating unanticipated changes. For example, in the case of the visitor design pattern, although new visiting behaviour can be easily accommodated for a given and fixed hierarchy or structure of visitable objects, the is no obvious way how to accommodate behavioural changes when the hierarchy is altered, due to the interrelationships that exist between the object hierarchy and the visiting functionality.

Subject Oriented Programming

Based on the principle of separation of concerns, subject-oriented programming advocates that software concerns "live" in a multidimensional hyperspace [Harrison and Ossher, 1993] and that it could be possible to create software by recognising and combining concerns in any of these dimensions. This is the case for anticipated concerns such as providing logging functionality to an existing application but does not hold when unanticipated ones come to play.

Aspect-oriented programming

Aspect-oriented programming [Kiczales et al., 1997] distinguishes aspects from components, which are units of systems' functional decomposition. Aspects are well-modularised concerns that crosscut a system's implementation and behavioural landscape. The primary goal of aspect-oriented programming is a) to cleanly separate components and aspects and b) to provide mechanisms for their abstraction and composition leading to the construction of an overall system. Aspects can be considered as behavioural adjustments, however the address fine-grained functionality like synchronisation, persistence or logging. Aspect-oriented programming does not address, yet, behavioural adjustments that may lead to the transformation of an object's behavioural landscape in a given context. For instance, it is not clear in aspect-oriented programming how to provide the necessary aspects in order to behaviourally modifying a component, say *person*, into a doctor. Furthermore, up to date, there are no dynamic aspect weavers a fact that requires re-weaving an entire application when behavioural changes occur.

Service-oriented composition

Service-oriented composition [Loverdos et al, 2002] is a paradigm that attracts many researchers. The idea is that a system is separated into services, which may co-operate, and such services may be composed or decomposed according to emerging needs. New services may be added and redundant ones may be removed without disrupting such a system. Service-oriented composition finds its root in behavioural composition. This is the only similarity between the proposed approach and service-oriented composition. The two approaches are in all other aspects different. In our approach business functions are modelled as classes and roles and their composition results to the required system function as opposed to service-oriented composition where the fundamental building block is a service. In this light, the ATOMA framework provides a more fine-grained vehicle for accommodating behavioural evolution, which however is not less expressive than is service counterpart.

CONCLUSIONS

In this work we presented an example of a tailorable e-government information system that supports behavioural evolution in order to manage unanticipated behavioural changes. The approach taken to support behavioural evolution is that of the ATOMA framework, where classes and roles are employed for this purpose. Through a simple yet appropriate example we show that not only unanticipated changes within a localised environment can take place, but also that between information systems that need to co-operate. The use of the role concept as a behavioural adjustment seems to be more appropriate than other proposed approaches since it can model both fine and coarse-grained changes in an object's behavioural landscape and it does so in a dynamic manner. The latter is considered to be the main advantage of role base evolution since changes happen as they occur and during the runtime of an information system.

REFERENCES

C. Alexander (1979), *The Timeless Way of Building*, Oxford University Press

B. Benz and J. Durant (2003), *XML Programming Bible*, John Wiley & Sons

D. N. Chorafas and H. Steinmann, (1993). *Solutions for Networked Databases: How to Move from Heterogeneous Structures to Federated Concepts*, Academic Press

E. Gamma, R. Helm, R. Johnson and J. Vlissides. (1994) *Design Patterns. Elements of Reusable Object-Oriented Software*. Addison-Wesley.

W. Harrison and H. Ossher. (1993). Subject-oriented programming (a critique of pure objects). *Proceedings of the Conference on Object-Oriented Programming: Systems, Languages and Applications*. Washington DC. 411-428

G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C.V. Lopes, J.-M. Loingtier and J. Irwin (1997). Aspect-Oriented Programming. *Proceedings of the 11th European Conference on Object-Oriented Programming (ECOOP)*, Finland, Springer-Verlag, LNCS 1241. June 1997.

D.S. Linthicum (2003). *Next Generation Application Integration: From Simple Information to Web Services*, Addison-Wesley Pub Co; 1st edition, 2003

Christos K.K Loverdos, Kostas Saidis, Anya Sotiropoulou, and Dimitrios Theotokis (2002), Pluggable Services for Tailorable E-content Delivery, In the Proceeding of the 8th International Conference on Object-Oriented Information Systems, OOIS 2002, Z. Bellahsene, D. Patel, C Rolland (eds), LNCS 2425, Springer

M. Lycett and R. Paul (1998a). Information systems development: The challenge of evolutionary complexity. *Proceedings of the 6th European Conference on Information Systems ECIS '98*, Marseille, France, 4-6 June 1998. Vol. I. 1-15.

M. Lycett and R. Paul (1998b). System Design as Ongoing Principle. *Proceedings of the Association for Information Systems 1998 Americas Conference*. August 14-16, Baltimore, Maryland. 968-970.

N.V. Patel and R.J. Paul (1998). Towards System's Tailorability: Comparative analysis of business change and system's flexibility. *Proceedings of the Association for Information Systems 1998 Americas Conference*. August 14-16, Baltimore, Maryland. 122-124.

N.V. Patel, L.A. Gardner and R.J. Paul (1995). Moving beyond the fixed point theorem with tailorable information systems. *Proceedings of the 3rd European Conference on Information Systems ECIS '95*, Athens Greece, June 1995.

J. F. Sowa. (1984). Conceptual Structures: Information Processing in mind and Machine. Addison-Wesley.

D. Stamoulis, P. Kanellis and D. Martakos. (2001). Tailorable Information Systems: Resolving the Deadlock of Changing User Requirements. *Journal of Applied System Studies*. 2/2.

D. Stamoulis, D. Martakos and L. Introna (1998). Systems for users, not for observers: The tailorability concept. *Proceedings of the 6th European Conference on Information Systems ECIS '98*, Marseille, France, 4-6 June 1998. Vol. 3 1011-1024.

D. Stamoulis, D. Theotokis, D. Martakos, G. Gyftodimos. Ateleological Development of "Design-Decisions-Independent" Information Systems. In Adaptive Evolutionary Information Systems, Nandish Patel (Editor), Idea Publishing Group, 2003. ISBN 1-59140-034-1.pp 81-104

D. Theotokis (2001). *Object-oriented development of Dynamically modifiable Information Systems using Components and Roles*, PhD Thesis, Dept. of Informatics and Telecommunications, Univ. of Athens, Sept. 2001 (in Greek)

D. Theotokis (2003). Approaching Tailorability in Object-Oriented Information Systems through Behavioural Evolution and Behavioural Landscape Adaptability, Accepted for publication in the *Journal of Applied Systems Studies Special Issue on "Living, Evolutionary and Tailorable Information Systems: Development Issues and Advanced Applications"*.

D. Theotokis, G. Gyftodimos, P. Georgiadis, and G. Philokyprou (1997). ATOMA: A Component Object-Oriented Framework for Computer Based Learning, in *Proceedings of the 3rd International Conference on Computer Based Learning in Science (CBLIS '97)*, G5-G15

FOOTNOTES

1 The term class is used in compliance with the notion of class in the traditional object-oriented paradigm, that is a construct to represent a real world entity.