Building a Virtual One-Stop Public Administration: from Users Requirements to a Conceptual Model

Olivier Glassey

BUILDING A VIRTUAL ONE-STOP PUBLIC ADMINISTRATION: FROM USERS REQUIREMENTS TO A CONCEPTUAL MODEL

Olivier Glassey[1]

[1] *INFORGE and Swiss Graduate School of Public Administration, University of Lausanne, Switzerland*
olivier.glassey@idheap.unil.ch

Abstract. *In this paper we demonstrate the methodology we used to build a model of a one-stop public administration. We began with a field study, studying a large public administration and surveying citizens. Having gathered field data, we processed it using summary cards and class descriptions that allowed us to show the types of services an administration can provide and to illustrate different categories of clients. Then we developed a conceptual model that covered both the structural and behavioural aspects of a one-stop public administration, using UML class models and use cases. During our work we realised the importance of having a good model in order to define detailed functional specifications and to develop a satisfying system.*

1. Introduction

This paper explains the steps we went through in order to build a model of a virtual one-stop public administration, from gathering users requirements, both amongst civil servants and citizens, to building a conceptual model. We believe there is a need for conceptual methodologies in that field: various researches are being conducted, such as the GAEL (Guichet Administratif En Ligne) project in Switzerland [5] or the BTÖV (Bedarf für Telekooperation in der Öffentlichen Verwaltung) method in Germany [8]. Indeed a stable model of processes can help public administrations managing the complexity and the rapid evolution of new technologies. It has to be noted that a Swiss working group on e-government appointed by the federal government has made similar remarks [9]. Furthermore this group identifies the definition of patterns for such applications as one of three essential action domains for e-government.

First of all we studied administrative processes in a large public administration in Switzerland, for there was a need to clearly identify and normalize all these different processes (internal processes, relations with its clients, services, procedures, etc.) before it was possible to dematerialise them [2]. We spent six months in the "Administration Cantonale Vaudoise" (ACV) and we met project managers, domain managers, departmental managers and users representatives for a total of 28 formal interviews. We also had many casual talks with different civil servants. This field research allowed us to collect a large amount of data on administrative processes, but it was rather heterogeneous since we met various people with very different focuses. We had to find a way to process that data and we will see below how we managed to do it. Moreover we conducted two online surveys (1998 and 2000) in order to discover the expectations of the citizens in the field of electronic administrative services and we had close to 500 questionnaires to analyse.

2. Specifying users requirements

After the interview stage, we filled in what we called summary cards to begin with our iterative modelling process. These cards are inspired by CRC cards (Class-Responsibilities-Collaborators), which were developed by Kent Beck and Ward Cunningham, two Smalltalk developers at the research labs of Tektronix in Portland Oregon. CRC cards constitute an informal class modelling technique that has some distinct advantages over simply starting with UML class diagrams [10]. For detailed information on this technique, we recommend [3]. These cards helped us collecting the input from the various people we met and transforming it into an object model through an abstraction process. For that model we focused on the services that the ACV delivers to its different types of clients, through different intermediaries or partners.

Throughout this paper we will use a simplified example of public health inspection to illustrate the iterative abstraction process we went through. *[Fig. 1]* shows a summary card for this service.

Service name	**Public health inspection**
Short service description	The department of Public Health has to inspect restaurants, shops, enterprises, etc. to verify if they comply with public health standards. It makes routine controls where the clients are aware that they will be checked and it also sets up unexpected controls. The results are made public.
Supplier	Department of health
Clients	Shops, restaurants, enterprises, medias, citizens
Uses types	Initiation, termination, notification, validation, publication, modification, consultation, search
Intermediaries/ partners	Consumers association, laboratories

Figure 1: Summary card

With the data gathered during our field research we prepared a little more than 90 summary cards, although we did not go into details for each of these administrative services. From these cards it was fairly straightforward to define class descriptions, such as shown in *[Fig. 2]*. We settled for 65 types of services that an administration can deliver to its clients and classified them according to the widely used typology of Information-Communication-Transaction services [1]. We also defined 50 types of clients that we will present below.

Service class	**Public health inspection**
Super-classes	Information - Communication
Sub-classes	
Methods	Initiated - Terminated - Notified - Validated - Published - Modified - Consulted - Searched
Attributes	Name - Description - Key_words - Supplier - Clients - Intermediaries - Supports - Payment
Constraints	
Collaborators	InstitutionalCl - BusinessCl - InternalCl

Figure 2: Class description

3. Building the conceptual model

We worked on this model using object-oriented (OO) technology and the graphical notation language UML (Unified Modelling Language) and we discussed this choice in details in one of our working paper. Here we assume that the reader has good notions of OO technology, but further reference on UML can be found in [4]. This object-oriented notation language has a very wide base of users, is an Object Management Group standard and combines the strengths of various object methods. It offers nine types of diagrams, amongst them the use cases diagrams that are very useful to formalize users needs. Besides the notation has been extended to model Web applications [7].

Such as stated in [4], we build models so that we can better understand the system we are developing. Indeed a good model allows us to specify the structure and the behaviour of a system, that is a static and a dynamic view of the system. In order to build the structural blueprint we used the class descriptions and the summary cards; we also worked on scenarios and use cases for the creation of the behavioural model.

3.1. Structural model

The structural model is made of class and object diagrams that show their attributes, their methods and the relationships between them. To build the hierarchy of the class diagrams from the class descriptions and the summary cards, we began with a fairly simple base diagram *[Fig. 3]*. A client requests a service from an administration and the supplier of this service can call different intermediaries if necessary, before delivering the service on whatever support suits best.

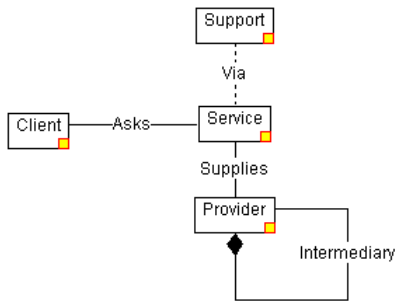

Figure 3: Base diagram

In order to build these class diagrams we used a CASE tool that allowed us to create cascading diagrams. That means that users can navigate through the class hierarchy by a simple click of the mouse. It is out of the scope of this paper to show the operations and the attributes for all the classes, however we

designed two synthetic diagrams showing the services and the clients hierarchy *[Fig. 4 and 5]*. Furthermore we created an index of over a hundred themes that can be used to classify services.

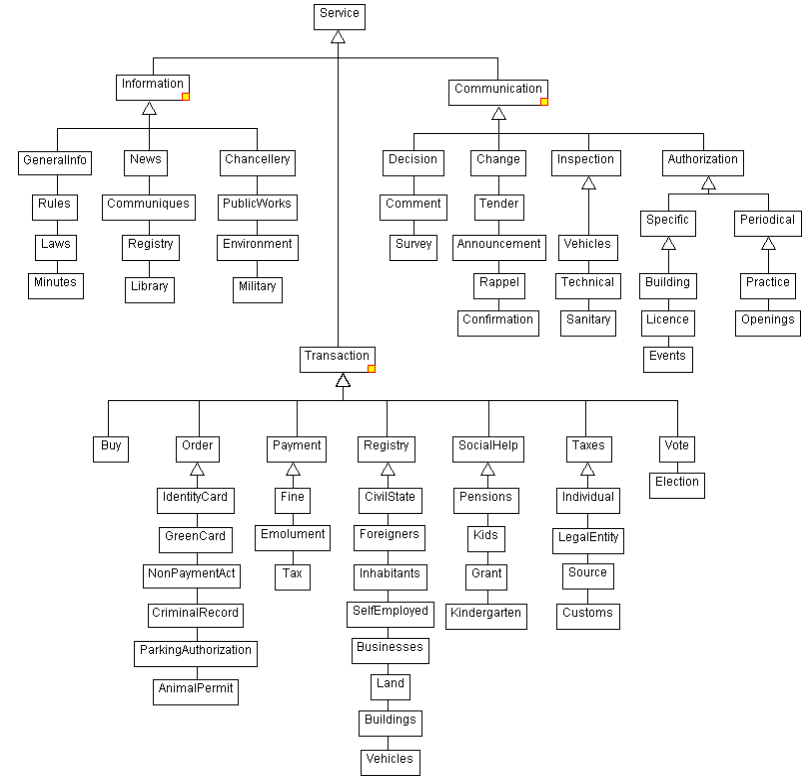

Figure 4: Typology of services

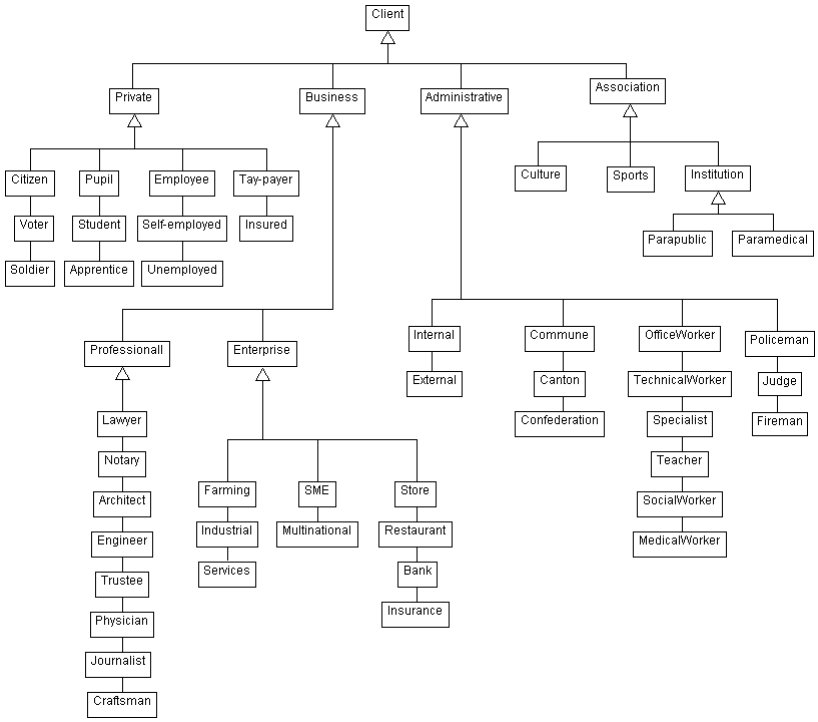

Figure 5: Typology of clients

We also identified the actors of the service delivery: suppliers, clients, intermediaries, etc. Then we established a typology of the suppliers: we decided to divide them into 6 general functions and 28 departments or offices. These divisions do not correspond to a real administration structure, but we think this functional visualization can well be used to represent real administrations with very different organizational structures. Last we defined the ways services can be provided (synchronous, asynchronous) and the support to be used: mail, fax, electronic delivery, face-to-face, etc.

Here we really want to emphasize the fact that this class model represents one point of view of a public administration in Switzerland and might not be of direct use elsewhere. However it can be modified or extended at any point and one does not have to use it as a whole. Although we realize this is only a partial representation of the reality and do not think that this is exhaustive, we believe that it does a good job covering what roles a public administration has to play in a modern society and that it constitutes a good backbone for our model. Furthermore we tested this typology with data collected by the State of Tennessee [12] and by the English government [6], who both conducted large-scale survey on Electronic Services Delivery. We were able to relate almost all of these Electronic Services to our typology.

3.2. Behavioral model

Once we had the structure of our system, we needed to define the order in which events occur in the system that is how and when messages are passed between objects, and how a specific class changes as it receives and sends messages. The core of our behavioural model is based on use case diagrams: they describe the behaviour of a system without going into implementation details.

We elaborated different scenarios (or textual use cases descriptions) from the summary cards, but we did not create use cases for each of the services we identified nor for each client, because it would have been too large a task and because we believed that going into to much details was not really useful for our model. Rather we developed a set of use cases that should form a good basis for anyone working with our model, thus allowing one to expand it easily by using a generalization/specialization hierarchy of services. The use case diagrams not only describe the behaviour of a system but also provide a common understanding of a problem between end users and domain experts. This common understanding is a very important condition to be able to develop, refine and validate the architecture of a system. Staying with our example of Public Health Inspection, *[Fig. 6]* shows how an employee from that department notifies a shop owner of a routine control at a certain date. The shop owner

acknowledges the control and receives the visit of an inspector. The latter makes the inspection and publishes a report that has to be validated by the manager before it can be published and consulted by the public or by internal clients.

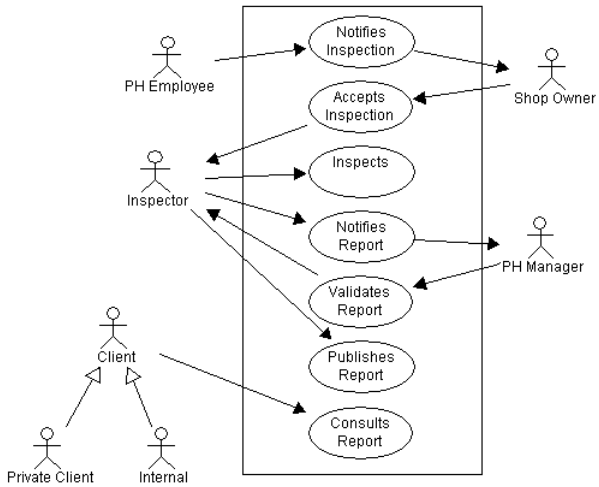

Figure 6: Use case diagram

Use cases diagrams are fairly simple and they provide an excellent mean for discussion between technicians, users and managers. Although they seem really trivial, they constitute a solid ground for building advanced scenarios. These show the normal sequence of events and alternate sequences. They also contain other information such as preconditions and postconditions, exception handling, triggers, etc. although not all of these fields need to be completed. The creation of these scenarios is based on an iterative process, as they need to be validated and refined. We show an example of such a scenario in *[Fig. 7]*.

Use case name	**Public Health Inspection**
Abstract	A shop owner is notified that the Public Health Services will inspect him. After the inspection, the inspector publishes a report that has to be validated and that will then be publicly available.
Normal sequence of events	1. A Public Health employee notifies a shop owner. 2. The shop owner accepts the inspection. 3. An inspector visits the shop and controls it. 4. The inspector writes a report and submits it to his manager. 5. The manager validates the report. 6. The inspector publishes the report. 7. Citizens or other employees of the public administration can consult the report.

Alternate sequence	2. The shop owner can postpone the inspection if he has valid reasons. 5. The manager does not validate the report.
Exception handling	5. If the manager does not validate the report, a specific procedure is undertaken.
Triggers	Inspections are normally routine based, but they can also take place after a complain.
Assumptions	
Preconditions	The shop has been single out for inspection.
Postconditions	A report is published.
Authors and date	Olivier Glassey, August 27th, 2001

Figure 7: Example of a scenario

Having defined the behaviour of a subsystem with use cases and scenarios, a specialist can specify the dynamic aspects of a model, represented by interaction diagrams. These allow a developer to make the functional specifications of a system application, by defining its objects and their roles, the messages passed between them and the sequencing in which the messages are passed. Interaction diagrams are of two forms: sequence diagrams that show the time ordering of messages and collaboration diagrams that emphasize the structural organization of the objects. Both forms are semantically equivalent and it is possible to convert from one form to the other.

4. Results

In the course of the development of the model, from the field research to the class diagrams and the use cases, we were confronted to several problems, but the main one was to find a good level of precision. That means we had to reach a compromise between degree of detail, abstraction level and simplicity of use: we did not want the model to be overblown with details, yet we wanted it to be generic enough to be used by different types of administration (federal, states, cities, departments, etc.) and to be simple enough to be understood not only by developers but also by users and decision-makers. There are different reasons that made this compromise difficult to reach. First, public administrations offer a wide range of services of all kinds: the English government has identified more than 500 of them [6]. Second, different kinds of clients have very different expectations: citizens, businesses, specialists, institutions, etc. And last, information and existing processes inside administrations are so heterogeneous that it can be very difficult to find a standard framework.

To face these loads of information, we found that the use of summary cards was really helpful. It helped us determine what was substantial and what was not, and in conjunction with UML it facilitated the creation of the class model and of the use cases. However, since value is not in the method, it is in the people that use the method [11], we could probably have had similar results using different techniques and methodologies. We now have a good conceptual model that we are using to define the functional specifications of a one-stop public administration system and we believe it is a great help for the development of a prototype. Furthermore we think that the model could be used by different types of administrations to build their own services. Certainly administrations would not use the model just as it is, but it can provide them a good basis to define and implement one-stop services.

5. Conclusion

During our investigation in a real public administration, we found that it was necessary to have a common language for the people concerned: users, managers, developers, decision-makers, etc. Without that common understanding, we think that a project is very unlikely to be achieved. The use of summary cards and of UML gave us a good ground for communication and collaboration between these persons. Furthermore, we think that these techniques successfully help bridging the gap between the reality and its representation, that is the model. Indeed they help us creating the structural and the behavioural models of the real organization.

To us, such a conceptual model is necessary because it allows its users to visualize a system as it is and to define it such as it will or should be. Then the model gives the developers a template to define the functional specifications of a system, independently of hardware or software. Once they have these specifications they can use CASE tools to generate the backbone of their applications and to facilitate their maintenance. We definitely think that a stable model of processes is very important in a world where technologies change so rapidly and where technological choices can have crucial consequences.

6. References

[1] Arthur Andersen. 2000: Egouvernement: Réflexions sur l'utilisation des nouvelles technologies de l'information et de la communication par les collectivités publiques. Arthur Andersen Business Consulting, Geneva.

[2] Baquiast, J.-P. 1999: Le guichet unique dans l'administration française. In New interfaces between Administration and Citizens, One-stop-government through ICT. International Workshop, Bremen, Germany.

[3] Bellin, D., Suchman Simone, S. & Booch, G. 1997: The CRC Card Book. Addison-Wesley.

[4] Booch, G., Rumbaugh, J. & Jacobson, I. 1999: The Unified Modeling Language User Guide. Addison-Wesley.

[5] Chappelet, J.-L. & Le Grand, A. 2000: Towards a Method to Design Web Sites for Public Administrations. IFIP Working Conference on Advances in Electronic Government, Zaragoza, Spain.

[6] CITU. 2000. Electronic Service Delivery: Spring 2000 Report. Central IT Unit, Cabinet Office, United Kingdom.

[7] Conallen, J. 2000: Building Web Applications with UML. Addison-Wesley.

[8] Gräslund, K., Krcmar, H. & Schwabe, G. 1996: The BTÖEV Method for Needs-driven Design and Implementation of Telecooperation Systems in Public Administrations. Universität Hohenheim, Stuttgart, Germany.

[9] GSCI. 2000: Guichet virtuel: la communication électronique avec l'administration, le Parlement et les tribunaux. Groupe de Coordination Société de l'Information, Switzerland.

[10] Harmon, P. & Watson, M. 1998: Understanding UML: The developer's guide with a Web-based application in Java. Morgan Kaufmann.

[11] Muller, P.-A. 1997: Modélisation objet avec UML. Eyrolles.

[12] STISP. 1999: Electronic Service Delivery, 1999-2000. State of Tennessee Information System Plan, USA.