A WEB-BASED ARCHITECTURE FOR E-GOV APPLICATION DEVELOPMENT

Marcelo Tilli, André M. Panhan, Osman Lima and Leonardo S. Mendes
Communication Networks Laboratory, Communications Department, Eletrical and Computer Engineering Faculty
University of Campinas, UNICAMP- Cidade Universitária "Zeferino Vaz"- Barão Geraldot, Campinas, SP, Brazil

Keywords: Electronic Goverment, e-gov, web-based system, MVC, Hibernate, Struts, Service-oriented.

Abstract: In this paper we present a web-based architecture for e-gov application development. We propose to study and characterize a software development environment of "electronic governance in cities", with emphasis on: legacy system integration, heterogeneous data base integration and development of an integrated web-based environment for the provision of e-gov services. This architecture is based on a 4-layer MVC model, the typical MVC (Model, View, Controller) plus a data layer, this architecture imposes few requirements to its clients, mainly a HTML web-browser, which allows this architecture to be platform-independent.

1 INTRODUCTION

A great deal of the development effort of software systems is centered on modeling, characterization and development of business models for the solution of specific problems. However, when you have to deal with the development of e-gov systems, particularly when the systems are built in an integrated network environment, there is a great dependency among all the systems that compose the general solution of e-gov applications.

Systems of people registration, services, taxes, protocol, finance, are common on a government environment. The problem is that usually, several different vendors, which use different platforms and different development tools, often build these applications. Then we end up with several heterogeneous and independent data bases, which are often incompatible. This scenario is very common on e-gov systems around the world.

On the last ten years, several countries have sponsored e-gov systems and projects of system interoperability. Initially, these architectures were created to access specific problems and not with the objective of being a generic model that could be universally accepted.

There are many research areas on the topic of electronic governance systems (Peristeras, Tarabanis, 2004), (Narasimha, Kumar, 2003), (Murthy, Kumar, 2003), (Song Gang, 2005), (Lenk, Traunmüller, 2001). The study of semantic information processing platforms, the development of interoperability platforms and legacy system convergence, are some of the many research areas that the study of advanced techniques of software engineering, computation engineering and communication engineering can contribute to solve several difficulties that faces e-gov system's software developers.

In this paper we present a web-based architecture for e-gov application development. We propose to study and characterize a software development environment of "electronic governance in cities", with emphasis on: legacy system integration, heterogeneous data base integration and development of an integrated web-based environment for the provision of e-gov services.

2 PREVIOUS WORKS

The work of (Peristeras, Tarabanis, 2004). This paper summarizes a broad research modeling effort, which aims at developing a domain description for the overall governance system. The authors propose the Governance Enterprise Architecture (GEA) as a set of domain models that serve as a top-level enterprise architecture. Namely, they present the mega-process model, the interaction model, the public policy formulation object model, the service provision object model and the latest development of the object model for the overall governance system.

Tilli M., M. Panhan A., Lima O. and S. Mendes L. (2008).
A WEB-BASED ARCHITECTURE FOR E-GOV APPLICATION DEVELOPMENT.
In *Proceedings of the International Conference on e-Business*, pages 476-479
DOI: 10.5220/0001911604760479

The work of (Narasimha, Kumar, 2003) propose an architecture for model development, for the planning, design and implementation of e-gov systems. The proposed model has four conceptual layers: Business Process Architecture; Data Architecture; Application Architecture; Technology Infrastructure.

3 SYSTEM ARCHITECTURE

We propose an architecture for e-gov systems for cities, that can access the general conditions of such systems, such as: the existence of legacy software, redundant, heterogeneous data bases and the paradigm shift from client-server platforms to web-based, integrated system for e-Gov services.

3.1 Architecture

For the architecture development, we used a modified MVC model (*Model-View-Controller*).
In pursuit of greater security and flexibility, we separated the client's communication layer from the business logic; hence we created an application layer responsible for dispatching requests and controlling its flows. The 4-layer architecture (*Model-View-Controller-Data*) is show on Figure 1 (Fowler, 2003).

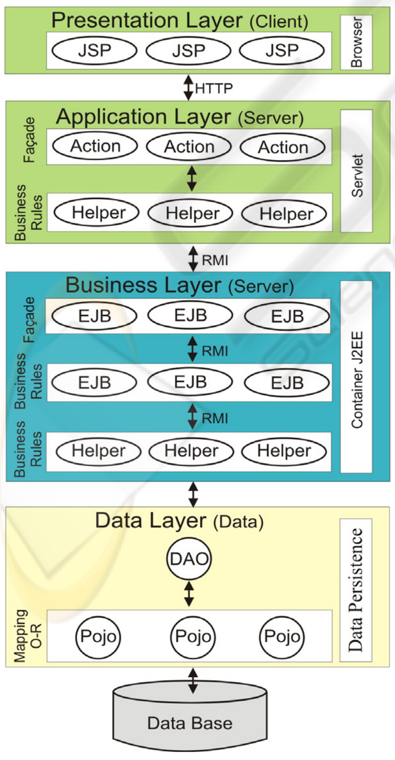

Figure 1: 4-Layer architecture.

This 4-layer model allowed us to reach some objectives: Facilitates the development of a distributed system, with security and with high availability; Enhances the automatization capacity of the city's public administration sectors, offering online services to the citizens, expediting its execution and accomplishing greater overall quality.

This architecture imposes few requirements to its clients, mainly a HTML web-browser, which allows this architecture to be platform-independent. With this architecture, there's no need to install/update the application software on the user's computer. Modifications of the system's software are made directly on the main server, which is transparent for the user. This is an improvement over the 2-layer model, where all update would have to be physically installed in the user's computer (Fowler, 2003).

The 4-layer model is as follows:

3.1.1 Model (Business)

This layer contains all the business logic and manages this logic through the use of EJB components. Therefore, this layer provides a structure for the implementation of distributed applications, aiming for the separation of concepts and to improve on quality factors, such as modularity, extensibility and reusability (Alur, Crupi, Malks, 2001).

3.1.2 View (Presentation)

This layer renders the model into a user interface element. It access enterprise data through the Model and specifies how that data should be presented. We are developing this layer with JSP (Java Server Pages) and JSF (Java Server Faces)

3.1.3 Controller (Application)

This layer processes and responds to events, typically user actions, and may invoke changes on the model. Based on the user's interactions and the outcome of the model actions, the controller responds by selecting an appropriate view. To facilitate the model's utilization, this layer should be able to interact with default Data-Base access technologies, such as JDBC and EJB and also be able to interact with third-parties technologies such as Hibernate, iBATIS, or Object Relational Bridge.

3.1.4 Data

This layer is the domain-specific representation of the information on which the application operates. It is responsible for the DAO (Data Access Objects), or the direct Data-Base access. This model allows for a transparent persistence of the objects models.

The data persistence allows us to aggregate several features to the proposed architecture, like the following: Transitivity persistence; transparent retrieval of data-objects (on a given object graph);

The purpose of the data-layer is to allow for an intelligent way to store object-oriented data, allowing for the convergence of heterogeneous data-bases.

4 AN E-GOV SYSTEM FOR CITIES

On this section, we will present a use case for a complete system for cities management, which can manage all services, citizen records, processes management and relevant data for a city's administration. The main purpose of this system is to provide consistent information to the manager to make the right decisions. From this prerogative, emerged the SIGM (Integrated System for Municipal e-Gov).

The application integration under a same domain was possible with the development of a single data base, covering, for example, citizens, business, social data, departments, processes, internal services, online services, systems users, addresses and others. Therefore, all data are concentrated and correlated under the same concept "Citizen Single Data Base" (BDUC).

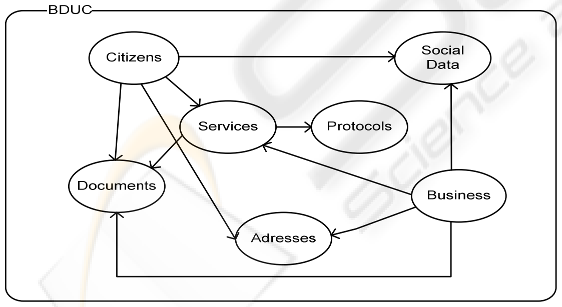

Figure 2: SIGM System Modules.

Through this concept, we achieved the interconnection between different modules of an e-gov system in an efficient and easy way, as show in

Figure 2. We have adopted the Hibernate technology for data base communication, which allows the independence of the DBSM (Data base System Management).

As this technology works with data object's concept, the conventional relational data base BDUC was created to facilitate the implementation and take advantage of all of the Hibernate resources. Therefore, we created a data base under two perspectives: object-oriented and relational, thus, it is possible to represent any data evolved with e-gov and its legacy systems.

This modeling, as in object oriented system, allows us to reuse data for creation of new tables, without affecting the system operation.

In particular for this use case, we used the Microsoft Windows 2003 Server operational system and the Oracle DBSM, but we can also apply the same use case to others platforms.

The SIGM can be deployed in any city, even those that have legacy application in use; since this kind of integration was planned by the adopted architecture and can be done in several ways:

- SIGM control the legacy applications through its functionalities and the BDUC data base.
- The legacy applications are incorporated by the SIGM.
- The legacy applications can use the business rules already implemented in the SIGM´s EJB layer.

After deployment, the SIGM can be accessed from any computer in any city location, by any person (citizen or government employee) registered in the system. These users can request certain services through the Internet, without the need to move to the city hall or any other service point.

This architecture also offers the possibility for application distribution on multiple servers, not only in the system execution, but also on its development. By distributing the development of a system of this magnitude, we obtain independence between the developer's teams and we can control more effectively each module implementation. By separating the business layer from the web layer, it is possible that others systems can use its functionalities without affecting the SIGM´s operation. This can be done by direct access of the EJB containers which are on the business layer.

To keep in order with the proposed architecture, on the web layer we adopted the Struts framework, which is an open source framework that works on MVC architecture and provides several control components to create dynamic Java web applications (Husted, Dumoulin, Franciscus, Winterfeldt, 2003).

This architecture also provided the development of an authentication system, which allows the SIGM to manage the permissions and profile control of the SIGM´s users and users of any other legacy application already in operation, as show on Figure 3. In these systems, all users and applications are registered through SIGM, and each one has its profile registered in different tables, so we can create different permissions profiles to different users for different applications. To control a legacy application, we just need to register the profile of this application and their permissions.

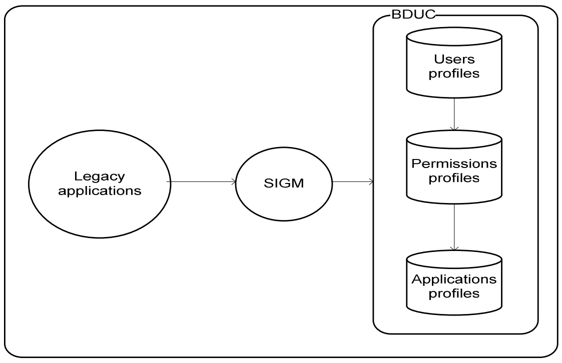

Figure 3: Legacy System Integration.

This use case was useful to demonstrate the efficiency of the proposed architecture and the technologies chosen for its implementation. In regard to this, we can consider that the use case was successful in its implementation.

The first version of the system was implemented and it is currently in use in the Brazilian cities of São José do Rio Preto and Campinas. We continue the system development, with new functionalities and features being added frequently, without hindering the system's operation.

5 CONCLUSIONS

This paper presented an architecture (SIGM) for the development of e-gov systems, taking in consideration government environment such as: the existence of legacy systems, redundant and heterogeneous data bases, the paradigm shift toward web-based platforms, the necessity of constant services development by the government, etc. The architecture is based on the MVC model with de addition of a data layer.

The SIGM purpose is to integrate heterogeneous applications under a distinct domain, to develop a single data base, that encompass data from all the citizens, enterprises, social, government departments, government procedures, internal services, online services, system's users, address, etc.

By using this single data base, all the data are converged and correlated under the concept of the BDUC, which data access is made through Hibernate, which allows the independence from a DBSM. SIGM can also control legacy applications through its functionalities and the BDUC data base.

REFERENCES

Peristeras, V., Tarabanis., 2004. Governance Enterprise Architecture (GEA): Domain Models for E-Gorvernance. In *ICEC'04, Sixth International Conference on Electronic Commerce, ACM.*

Narasimha M., Kumar P. R.V. 2003. Software architectural design model for e-Governance systems. In*, TENCON 2003. Conference on Convergent Technologies for Asia-Pacific Region.*

Murthy, N., Kumar P. R.V., 2003. Software architectural design model for e-Governance systems. In *TENCON 2003. Conference on Convergent Technologies for Asia-Pacific Region.*

Gang S., 2005 Transcending e-Government: a Case of Mobile Government in Beijing*, The First European Conference on Mobile Government, Brighton, July 2005*

Sánchez-Nielsen E., Chávez-Gutiérrez F., 2006. Smart Diffusion of Government Contents. In *ICE-B – International Conference on E-Business.*

Lenk K., Traunmüller, R., 2001. Broadening the Concept of Electronic Government. In *Designing E-Government, Prins J.E.J. (ed.), Kluwer Law International, pp. 63-74.i*

Fowler, M., 2003. *Patterns of Enterprise Application Architecture.* Addison-Wesley, Inc.

Alur, D., Crupi, J., Malks, D., 2001. *Core J2EE Patterns*, Sun Microsystems, Inc.

Husted, T., Dumoulin, C., Franciscus, G., Winterfeldt, D., 2003. *Struts in Action.* Manning Publications Co.

Bauer, C., King, G., 2005. *Hibernate in Action.* Manning Publications Co.