Architecture to Application Gateway to access Electronic Government Legacy System

Pedro Luiz Pizzigatti Corrêa
Escola Politécnica da Universidade de São Paulo,
Av. Prof. Luciano Gualbeto, São Paulo-SP, Brazil
pedro.correa@poli.usp.br

Moacir Pedroso Jr
Empresa Brasileira de Pesquisa Agropecuária - EMBRAPA,
Brasília-DF, Brazil
moacir.pedroso@embrapa.br

José Martins Jr.
Escola de Engenharia de Sâo Carlos da Universidade de São Paulo,
São Carlos-SP, Brazil
jmartins@sc.usp.br

Abstract : In government organizations, it is common the existence of legacy system with the objective to preserve the functionalities and public investments made during many years, mainly within mainframe based architectures. With the evolution of the business rules and technology, arises the need to develop new functionalities and to integrate them with legacy systems and data. This paper approaches this problem by proposing a model of Legacy System Integration and Application Gateway Architecture. A case study of this architecture is presented with an implementation of an application gateway in an electronic government context.

1 Introduction

Throughout the years, all different instances of government have been developing autonomous Information Systems targeted to specific public administration domains. Large investments where made in the creation and in the evolution of these systems, due to the need to adjust them to a changing business rule environment as well as the technological infra-structure upgrade needed for its operation and administration.
Nowadays, we observe a sharp increase in the demand for more and more services integration, which requires information flow between existing processes between the government and the society as a whole, as well as intra-government. This situation imposes the need, to all levels and instances of government, to integrate their different information systems. In the case of legacy system, many difficulties arise because several of these systems where based upon proprietary architectures, especially those developed for Mainframes.
A common characteristic of government legacy information systems is the large amount of data that they process and store, developed with procedural languages, operating huge quantities of data stored on non-relations DB, and running in large mainframe computers. It's not rare the necessity to integrate such systems with Web applications, or even with other information systems running in different computational platforms. Certainly, this is not a

trivial task, condemning that many electronic government legacy systems are not available for widespread usage.
The usage of middleware, such as MQ-Series, CORBA, COM+/NET allow for the generation of environments that facilitate the interoperability of these systems, but a government institution may not always rely on the availability of such solutions due to the high acquisition costs and complexity of usage, demanding, then, a medium to long range planning, which may sometimes be in conflict with the necessity for immediate evolution of such systems.
Therefore, it must be emphasized the need for works that allow the migration of these systems to an open architecture, based on the following principles:

- Consider the existence of different application domains;
- Be flexible enough, as to allow for different government organizations to establish local strategies for adjustment of the systems, mainly towards the preservation of the investments already made and to the maintenance of the existing services in regular operation.

The objective of this work is to present an Application Gateway Architecture that considers the requirements previously stated, and to present a case study of the architecture, by means of an Application Gateway developed for the São Paulo State Finance Department (Brazil), that interacts with the São Paulo State Taxpayer's System, hosted in a Mainframe.
The sections of this article attempts to characterize the problem of accessing legacy systems, then present a study a Application Gateway architecture that incorporates the aforementioned requirements, and after presents a case study of an implementation of this architecture. At the end, the results obtained with the implementation, and the conclusions about this work are presented.

2 Related works: characterization of Application Gateway

Using the classification proposed by [Widerman97] for the phases of information system evolution within its life cycle, the problem of providing Web access to the functionality of legacy systems can be characterized as "modernization", as it deals with changes more substantive than those required by regular small interventions to support business rules evolution, but it is not a replacement of the current system – on the opposite, it tries to preserve as much as possible the functionality offered by the legacy system. The diagram in figure 1 illustrates this [Comella-Dorda 2000].

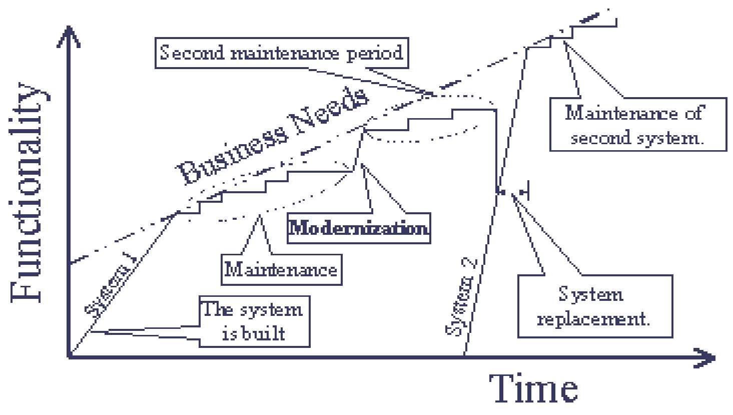

Figure 1. Information system evolution life cycle [Comella-Dorda 2000]

Depending on how much it is necessary to understand the legacy system code, a modernization can be characterized as "white box", when direct interaction with the implemented functions is required, or "black box", when this interaction happens only indirectly, through the system's input and output interface system [Widerman 97].

[Comella- Dorda2000], in a survey about modern approaches for system modernization, considers modernization under three aspects:

a) User interface modernization, with the purpose of providing access to legacy systems through a more modern interface (for example, graphical user interface (GUI), web interface (html), implemented usually by means of a "screen scrapping" technique;
b) Data modernization, to allow that new systems implemented using incompatible technologies may access data managed basically in the legacy system environment. This happens by the means of Database Gateway Integration, using techniques of Distributed Databases [Correa, 2002], and of connectivity tools such as ODBC and JDBC;
c) Functional or logic modernization, to embed directly the implemented business logic, with direct access to functions. CGI (Common Gateway Interface) Integration, Object oriented wrapping, as in CORBA, and Component Oriented Wrapping, as is DNA, CORBA 3, and EJB (Enterprise Java Beans) (Fremantle, 2002).

The approach adopted in this article cannot be exactly classified in any one of these categories because, as can be seen in the following section, the requirements considered for system evolution were very restrictive and could be only obtained by the implementation of new business rules by means of encapsulation of several legacy system transactions, composing a new unique transaction in the extension of the modernized system.

3 Requirements to Access the Government Legacy Systems

Under the viewpoint of information systems, it is possible to establish a diagnostic based on the analysis made of the São Paulo State Treasure Secretary systems, that is valid for many government institutions that are undergoing a modernization process.

There were identified many transactional systems, related to specific public administration functional areas, like Register, Tributary and Financial Administration, Payroll, Legal Documents Print Control, and so on; as well as new systems under development that integrate in a unique process several areas, like New Company Opening and Taxpayer Infraction Management through the WEB. Analyzing these systems, it is possible to classify them as follows:

- *Autonomous and Heterogeneous legacy systems*: in a broad view, the legacy systems are of self-contained, development over different technologies and using distinct computational platforms;
- *Demands for new systems:* originated from new business rules, as well by technology evolution necessity, or even by the need to automate manual processes;
- *Legacy applications reengineering*: evolution of several legacy applications demanded by technological changes, since they no longer match the new functional requirements.

Considering the necessity of services and information exchange between the systems, it is possible to define a simple Systems Integration Conceptual Model to allow for the technological heterogeneity and independence from legacy applications, in an shared and distributed application environment, and that could make easier to access to the legacy systems from the new applications. The model must possess the following characteristics:

- Use mature technologies and available tools;
- Be flexible to accommodate several technologies, allowing the evolution of the system,
- Preference for open tools supported by many software suppliers.

The Systems Integration Conceptual Model proposed in this work, has the objective to establish the fundamentals of Application Gateway Architecture, described in item 4. The concepts of this model are (see Fig. 2):

- *Domain*: represents a system or a remote database (local or external government unit),
- *Application Gateway*: each domain define the services to be offered, which are implemented in the Application Gateways ("Wrappers"),
- *Client Proxy*: each client implements a proxy in its own domain to concentrate the requests for application gateways services;

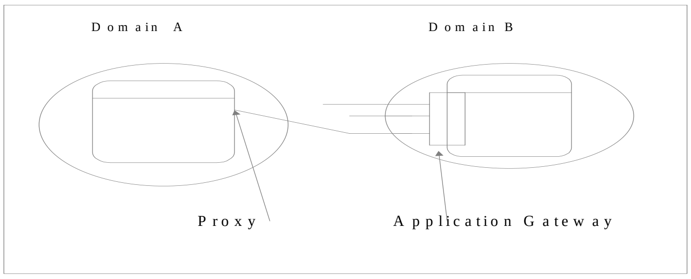

Figure 2: Elements of System Integration Conceptual Model.

The Application Gateway has the objective to allow the access to the services and information from a Legacy Systems Domain, owned by the government unit, isolating the data access logic used by Legacy Systems from the others corporation applications. This characteristic allows for the independent evolution of the legacy systems independently and isolated from other client applications.

It is necessary to say that the proposed Gateway Model doesn't define any especial business service, as these are identified accordingly to the legacy application domain, but there are some non-functional requirements of the services to be considered, like:

- Secure access to the legacy applications: the gateway has to consider the security mapping mechanism between different domains;
- Access performance to the Legacy Services: considering that the systems are in different computational platforms, it is possible to have excess of demand of the legacy services, impacting the desired performance. So it is necessary to establish parameters of service quality to restrict the number of requests as to guarantee the desired performance.

Is necessary to emphasize the importance of a simplified initial model for legacy system integration, because it can be by the Software Engineers during the process of development, as well as by the Business Analysts that are defining the integration between different domains and will support the definition the following requirements:

- What information and services that the client application need to access, the access profiles, the volume and source of the information;
- Data valid: what is the valid of the accessed data (volatile),
- Performance: identification of response time required to the Application Gateway in response to the client application.

Based on the requirements identified above, we can propose a Application Gateway Architecture that implements the Integration System Information Model.

4 Application Gateway Architecture

This section presents the Architecture of the Application Gateway that supports connectivity features to legacy applications.

As an initial requirement in the project, the Gateway must be designed to respond several requests from Applications simultaneously, through the Proxies, providing a transparent and distributed access to several resources. These resources are:

- Several types of databases, located on different servers or Mainframes;
- Legacy systems or applications, running on different operating systems.

There is another functionality of the gateway that concerns the quality of the provided services, as follows:

- Execution priority for a call performed to connecting layer, based on service and user identification;
- Synchronous or asynchronous service execution;
- Transaction atomicity, that is, while the execution has not been done completed (all operations), the transaction has not was not been executed;
- Data consistency on distributed databases;
- Configuration options for error handling, and reactions to low system resources.

It is opportune to add that the proposed Gateway Architecture resulted from the decision to use a n-layers model in opposition to the traditional client/server model. The architecture reflects the data distribution and the need to obtain certain quality parameters to the operations, executed on several environments.

Three layers were stated to the Gateway Architecture, as showed on Figure 3.

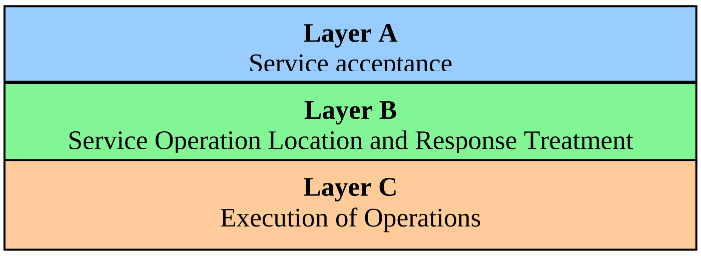

Figure 3: Architecture layers of the Application Gateway.

i) Layer A:

The Layer A is responsible for accepting the requested service and to manage several clients' requests, comprising the higher-level interface service treatment. This layer manages a service queue to be processed, implementing priority polices using the quality of service parameters. Besides the treatment of service requests, this level also implements the how the service will be handled, either synchronous or asynchronous, as defined in configuration parameters for the service.

ii) Layer B:

The layer B is responsible for locating the requested service and primitives operations that compose the service. Each primitive operation has a specific process that manages the primitive that could correspond a Mainframe transaction or a to a DBMS (Data Base Management System) operation. This layer is responsible for some functions of quality of service like:

- Transactional atomicity of service execution;
- Resource management for primitives operation execution;
- Data consistency in distributed databases;
- Primitive operations error treatment.

iii) Layer C:

The layer C is responsible for primitive operation execution. The primitive operation receives a request from layer B and returns the results of the operation. These primitives operation can be:

- Stored procedures of DBMS, activated by methods like ODBC, JDBC, etc.

- Access to mainframe applications, thought the screen interfaces using HLLAPI (High Level Language Application Program Interface), using communications primitives like APPC (Advanced Program to Program Communication), or other middleware like MQ-Series;
- Data transfer process, to access the information in remote domains, building a database cache.

Examples of use of this architecture are illustrated by two use cases depicted bellow. In Fig. 4, there is a sequence of operations related to access to a transaction in legacy system hosted in a mainframe. Fig. 5 depicts a typical use case for access to a DBMS.

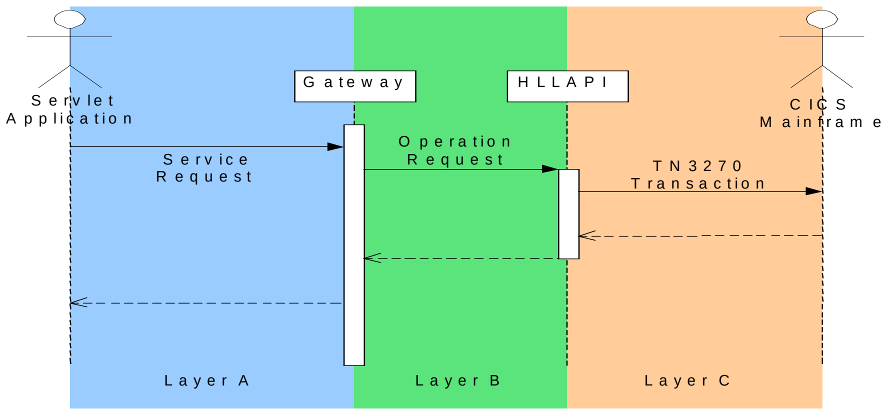

Figure 4 Use case to access a mainframe transaction

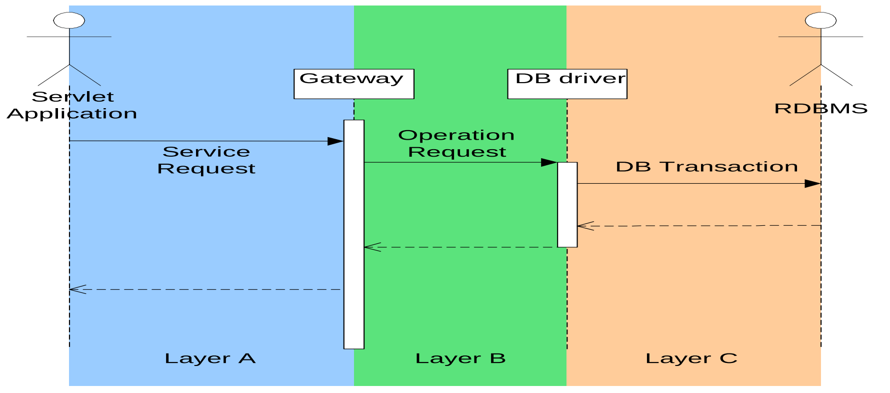

Figure 5: Use case to access a DBMS.

It is important to be noted that this architecture defines services to be implemented by the Application Gateway. Depending on the technology adopted in the unit of government, part of these services could be implemented directly and the operational environment available, in which the application gateway runs, could supply part. For example, considering that there is a middleware that support part of these services, like connection management and primitive localization, so the Gateway could be using this tool, and implementing only the services that are not available.

5 Case study of the proposed architecture

In what follows, it is presented a case study of this architecture, as implemented to access a mainframe legacy system. Figure 6 describes the used technologies.

In accordance with the proposed architecture for the case study, it can be seen that any application to be developed for Internet can access the application gateway services using the available proxies (DLL, in MS-Windows® environment and shared libraries, in unix-like environments). To extend the existing solution to other computational platforms, one must only develop the proxy for the specific environment, which is much simpler than the application gateway.

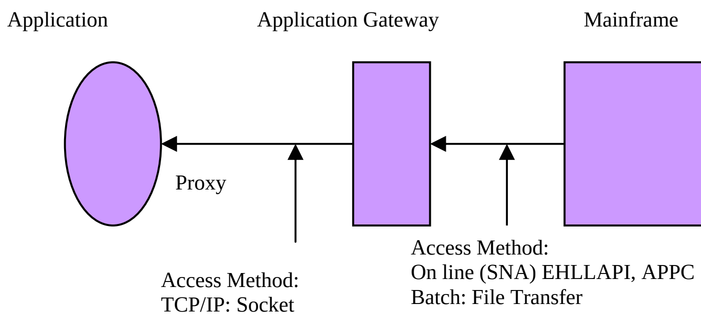

Figure 6: Case study of the proposed architecture

It must be observed that, depending on the quality of services requested by the application, the application gateway may select either the on-line access method or the mainframe batch execution of the requested service

In what follows, the actual implementation of the study case will be detailed following the Figure 7.

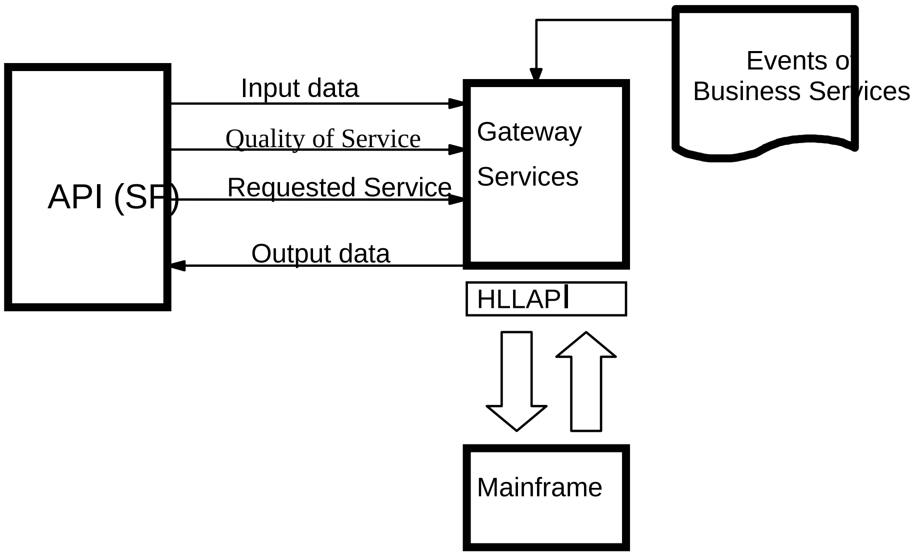

Figure 7. Case study of the implemented architecture.

The proxies are implemented by APIs (Application Program Interface) that access the Application Gateway. The API defines a set of parameters that are interpreted by the Application Gateway. For example, consider an application that handles taxpayers' data, where the service for inclusion of a new taxpayer requires access to a legacy system, through the Application Gateway. The parameters sent to the Gateway are:

- Taxpayer data;
- Quality of service: either synchronous or asynchronous;
- Requested operation: which service is being requested. In the case of the example, requested the inclusion of a new taxpayer

It must be pointed out that the technique available to access the mainframe is the HLLAPI, and therefore it is necessary to handle the user interface screens of that legacy application.

The representation of the mainframe interface screens, that indicates to the Gateway the path to follow to execute the service, is parameterized in a text input file named Events of Business

Services in Fig. 7. As this information is parameterized, it is relatively easy to maintain and evolve the gateway whenever the mainframe interface screens change, without the need to change the gateway code.

6 Conclusions

It is important to consider the use of Application Gateways in Electronic Government Information Systems Architectures, given the existence of legacy systems and the necessity to preserve previous investments and to maintain operational the functionality managed by these systems.
This work discusses an Integration Model for Legacy Systems and Application Gateway Architecture, based on the proposed model, with the objective to contribute to works related to the Electronic Government System Integration.
The proposed architecture can be implemented based on middleware solutions such as CORBA or MQ-Series, or using connectivity tools available natively in the operation environment to access the mainframe, relational databases, or even using TCP/IP technology.
The case study presented in this work shows a possible implementation of the architecture integrating government Information systems throughout different platforms and different application domains, based on Mainframes and web distributed applications.
A possible extension of this work is to incorporate SOA (Service Oriented Architectures) concepts (Papazoglou & Georgakopoulos, 2003) to the legacy system integration model, and elements from Web Services Architecture to the Application Gateway.

References

1. Baldonado, M., Chang, C.-C.K., Gravano, L., Paepcke, A.: The Stanford Digital Library Metadata Architecture. Int. J. Digit. Libr. 1 (1997) 108–121
2. Cornelta-dolta, S., Wallnau, K., Seacord, R.C., Robert, J. “A Survey of Legacy System Modernization Approaches” (CMU/SEI-2000-TN-003), Pittsburg, Pa. Software Engineering Institute, Carnegie Mellon University. Available WWW <URL: http://www.sei.cmu.edu/pub/documents/00.reports/pdf/00tn003.pdf>
3. Correa, P.L.P.. Diretrizes e procedimentos para o projeto de bases de dados distribuídas. (Doctorate Thesis) Escola Politécnica da USP, São Paulo. 2002, 198pp. Guidelines and procedures for design of distributed databases.
4. Fremantle P.; Weerawarana, S.; Khalaf, R.: Enterprise Services. Communications of ACM, v. 45 n.10, p.71-76, out. 2002.
5. Marco T., Letizia J., Carl-Fredrik S., Alf I. W. COTS Products Characterization, Proceedings of the 14th international conference on Software engineering and knowledge engineering, July, 2002
6. Papazoglou, M.P. & Georgakopoulos. Service-oriented Computing. Communications of ACM, vol. 16 no. 10, October, 2003.
7. Stal, M. Web Services: Beyond Component-Based Computing. *Communications of ACM***,** v. 45 n.10, out. 2002.p.71-76
8. Steve V, New features for Cobra 3.0, COMMUNICATIONS OF THE ACM.Vol. 41, No. 10 October 1998.pg44-52.
9. Weiderman, Nelson H.; Bergey, John K.; Smith, Dennis B.; Tilley, Scott R. Approaches to Legacy System Evolution. (CMU/SEI-97-TR-014). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1997. Available WWW <URL: http://www.sei.cmu.edu/pub/documents/97.reports/pdf/97tr014.pdf>