Architecting Frameworks for Specific Applications with RM-ODP

Ana Paula Gonçalves, Sandro Antônio Vicente, Dib Karam Jr, Moacyr Martucci Jr
Computing Engineering and Digital Systems Dept – Escola Politécnica da USP
apaulacg@uol.com.br, sandro.vicente@poli.usp.br, dib@poli.usp.br, moacyr.martucci@poli.usp.br

Abstract

Today, distributed systems are commonly used in business enterprises in practically all market sectors. But such systems are characterized by their huge complexity due to physical distribution, lack of synchronization, heterogeneity, external parties and the very business logic related to the system itself. RM-ODP appeared as an interesting resource to assist the designing of architectures for distributed systems, providing means to capture business needs, distributed processing systems architecture, semantics of processing, and choice of technologies. This paper presents three different experiences in modeling architectures using RM-ODP: a proposal for use of RM-ODP for the development of convergent applications, a generic architecture for CRM systems and a framework based on ODP for the integration among different computer architectures.

1 Introduction

Nowadays, distributed systems became mature enough to be commonly used in business enterprises in practically all market sectors. Besides their business complexity, such systems are characterized by physical distribution, lack of synchronization and heterogeneity due to the fact that they are composed of a plethora of different applications and devices. In addition, such systems are also affected by external elements, such as telecommunication networks, 3rd party systems, etc.

Hence, when a distributed system is being designed, it is necessary to assure that its architecture will provide the right levels of service, supporting interoperability, scalability, security, heterogeneity, and all other aspects that characterize such systems.

RM-ODP (Reference Model – Open Distributed Processing) appeared as an interesting resource to assist the designing of architectures for distributed systems, providing means to capture business needs, distributed processing systems architecture, semantics of processing, and choice of technologies, all in a consistent and complete manner. It focuses on how to capture the components, their interrelationships, and formal semantics of processing aspects of an open distributed processing system.

The paper objective is trying to show how the RM-ODP can be considered a complete modeling tool, presenting an overview for three generic cases. Each case focuses a different set ofviewpoints, chose according its importance in application. The complete set of the viewpoints were used considering the three presented cases together.

These experiences in modeling architectures addressing specific applications domains carried out by integrants of the Computing Engineering and Digital Systems Dept of Escola Politécnica da USP (University of Sao Paulo), which have been driving early researches in this area in Brazil.

This paper is organized as follows. Section 2 presents a proposal for use of RM-ODP aiming the development of convergent applications. Section 3 presents a generic architecture for CRM (Customer Relationship Management) systems, which fits CRM strategies into the ODP (Open Distributed Processing) enterprise language. Section 4 outlines a framework based on ODP for the integration among different computer architectures. Finally, section 5 concludes the paper.

2 Designing technologically convergent applications using RM-ODP

This section presents a proposal for use of RM-ODP aiming the development of convergent applications. But it is important first to clarify the meaning of the term “convergent”.

The term technological convergence is often mistaken for interoperability. The difference among them is that “interoperability” means capability to work with another autonomous systems or applications. Technological convergence means adaptation of services to different communications medias through the use of networks and terminals designed to bear such services, transparently providing users with access to these services’ information and applications. So these services are carried out using

any network, any communications channel and providing a coherent human-interface with appropriate quality. This demands fundamentally capabilities of mobility, portability of applications and content, and interconnection and interoperability among platforms and operators. Any application involving several technologies, such as: digital television, mobile Internet, videoconference, telephony, interactive broadcasting, etc, can be considered a convergent application [1].

An appropriate solution in order to develop convergent applications employs the model of distributed computing using the Internet, client-server architecture in a decentralized networked environment, and independent and autonomous devices. Generally, this distributed computing model uses mainly the TCP/IP (Transmission Control Protocol - Internet Protocol protocol, for the communication among devices, using different architectures, operating systems and applications. Distributed object architectures provides suitable means to apply the model of distributed computing, enabling the transparent integration of distributed services upon software architectures, hardware platforms and networks [2][3].

Figure 1 presents an overview of architecture of convergent applications based on three different technologies: the Internet, mobile telephony and digital television. The integration among these three technologies is possible only through the use of standardized protocols and distributed objects. Thus, next section presents how convergent applications can be developed using the RM-ODP.

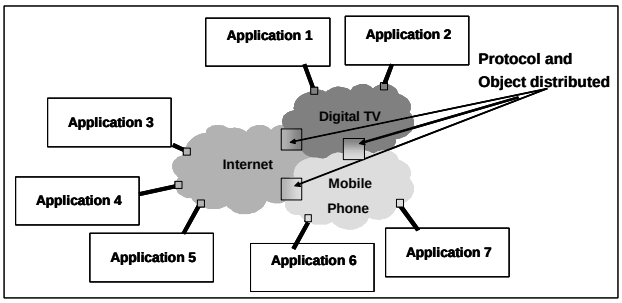

Figure 1. General view of architecture of convergent applications based on three different technologies

The following subsections present an overview of architecture for convergent applications using the computational, engineering and technology ODP viewpoints. The enterprise and information viewpoints are not addressed in this article because it aims to propose a generic architecture, able to be applied over different enterprise and information models.

2.1. Computational viewpoint

The computational viewpoint is responsible for the functional decomposition in terms of objects and their computational interfaces, which must be specified for the development of convergent applications [4]. Figure 2 presents the objects and interfaces and they are detailed as follows.

- User object: represents communication channel through which a user may use to send or receive information and services. Examples of instances of this object are: PDAs (Personal Digital Assistants), mobile telephones and digital television. This object has interface with Connection Object;
- Connection object: is the object responsible for connection services among the user object and the service required, according to the type of communication channel. Examples of instances of this object are: Internet connection via TCP/IP, mobile phone connection via WAP (Wireless Application Protocol), digital television connection using image, sound and data compression and decompression techniques through the use of DVB-T (Digital Video Broadcasting - Terrestrial) standard. This object has interface with Convergence Services Objects;
- Convergent services object: these objects represent convergent services, so they can be considered an application framework integrating different convergent services and obtaining services and information from different locations.

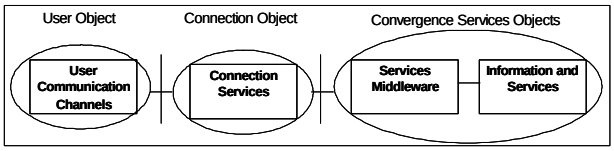

Figure 2. Computational view of objects for convergent applications

2.2. Engineering viewpoint

The engineering viewpoint model provides support for the execution of the computational model [4]. Figure 3 presents a simplified architecture in the engineering viewpoint, detailed as follows.

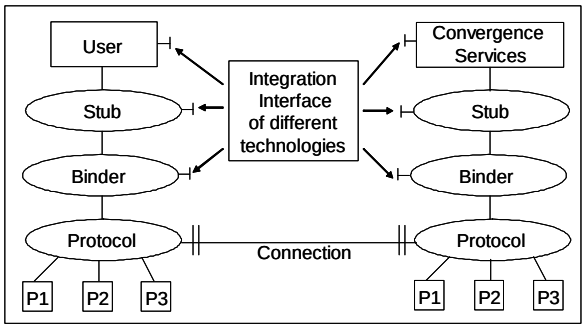

Figure 3. Simplified architecture of convergent applications in the engineering viewpoint

When an *user* object requires a service or information, a *stub* object will provide functions to support transparency in the request of a service and its response. The *binder* object verifies compatibility among interfaces and keeps the connection integrity of the service request and its response. The protocol object interacts with protocols of the other different convergent services. For example, in figure 3, protocols like P1, P2 and P3, transparently obtain the information or service from the binder and stub objects of the instances of the *convergent services* objects. The integration among Internet services, mobile telephony and digital television, for example, is possible through the use of interfaces concerning different technologies. An example for the use of this architecture would be a request of a service from a digital TV device (*user* in figure 3) that would enable a content search in a database server (*convergence service* in figure 3) using the Internet (*connection* in figure 3), whose answer would be delivered to the user's mobile phone (*user* in figure 3).

2.3. Technology viewpoint

The technology viewpoint specifies the software and hardware components of the application [4]. Such technologies for convergent applications were proposed in light of studies on well-grounded technologies, availability of tools to support development and ease of development.

The platform generally adopted is Java, because it provides APIs (Application Programming Interfaces) and facilities for the development of applications using different communications medias and devices, such as the following:

- Internet integration interface: J2EE (Java 2 Plataform Enterprise Edition) platform for the development of distributed objects allows both the use of an IDL (Interface Definition Language) with CORBA (Common Object Request Broker Architecture) and XML (eXtensible Markup Language) with SOAP (Simple Object Access Protocol);
- Mobile telephone integration interface: use of the technology MIDP (Mobile Information Device Profile). The main characteristics of MIDP concerning software are the use of specific Java APIs for limited devices, as for example: *java.lang.** classes, *java.util.** classes, *java.io.** classes, HTTP (hypertext transfer protocol) 1.1 protocols and HTTPS (hypertext transfer protocol security) X.509 protocols. Characteristics concerning hardware are the screen size of 96x64, 1-bit intensity, black and white, 4,096 colors or touch-screen enabled;
- Digital television integration interface: assuming the use of the European standard DVB-T, the MHP (Multimedia Home Platform) can be employed. Its main characteristics concerning software are the use of Java APIs like, for example: Personal Java, java TV API, Java Media Framework and DVB API extensions. Characteristics concerning hardware are MPEG-2 (Moving Picture Experts Group – 2) reception, screen resolution of 720x576 pixels and "true color" model.

Figure 4 presents the technology layers proposed for the development of convergent applications. The layer that represents the facilities of the applications' implementation is the main concern and is shown in the figure 4 with shaded blocks.

3 Designing a CRM architecture with ODP

In this section, an overview of a generic architecture for CRM systems is presented in light of the enterprise and information viewpoints. At first, the main idea of CRM concept is stated, as well as the concept of CRM Ecosystem [5], which is essential to drive the modeling of the CRM architecture in an enterprise context.

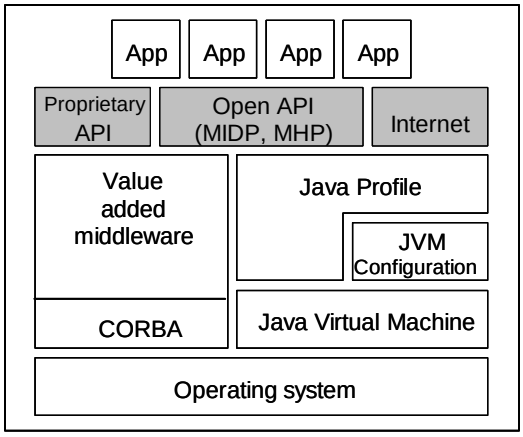

Figure 4. Technology layers for the development of convergent applications

3.1. Customer Relationship Management - CRM

CRM is a marketing concept whose goals are the acquirement of new customers and the loyalty of existing ones. These goals are reached establishing a friendship relationship with customers, employing one-to-one interaction, to achieve complete knowing of them, predicting their behavior and habits [6]. Technology supports one-to-one customer interaction by means of automated and semi-automated contact points providing accessibility and distribution of information to the customers [7]. Interactions among contact points and front-office applications (such as sales, marketing and customer services) implement CRM process.

CRM is implemented through the automation of horizontally integrated business processes involving front-office customer touch points via multiple, interconnected delivery channels [5]. We can distinguish three large functional groups necessary for a CRM architecture: *operational*, *collaborative* and *analytical*, where the operational CRM can be divided into front-office and back-office. Front-office functions are performed by customer service, marketing automation and sales automation applications. Back-office are performed by Enterprise Resource Planning (ERP) systems, Supply Chain Management (SCM) systems and legacy systems. Analytical CRM comprises decision systems and tools for business performance analysis such as: data warehouses, data marts and data mining tools. Collaborative CRM comprises elements used as customer channels, such as: IVR (Interactive Voice Response) devices, CTI (Computer Telephony Integration) systems, ACDs (Automatic Call Distributor), web sites, agents terminals, etc [6]. These three functional groups working together establish a CRM Ecosystem, which is depicted in figure 5.

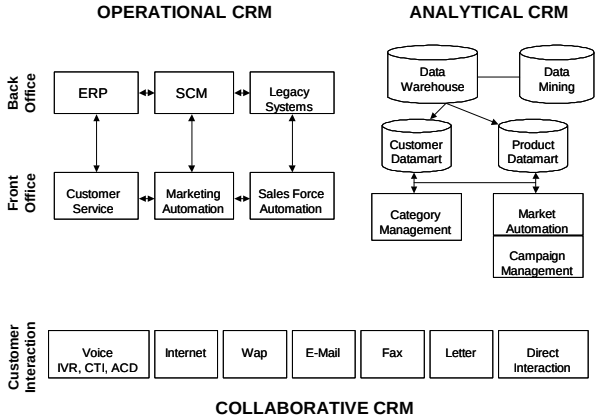

Figure 5. The CRM Ecosystem

3.2. Enterprise viewpoint

In this subsection, a generic architecture for CRM systems is modeled using the ODP's enterprise viewpoint [4]. This viewpoint is the basis to understand the overall structure of the CRM architecture in terms of which elements take part in the architecture and which roles they perform.

In the enterprise viewpoint, the entire CRM system is modeled by means of business objects, or enterprise objects, each one performing the roles necessary for the activities concerning those functional groups that perform the CRM process: *Collaborative* (CO), *FrontOffice* (FO), *BackOffice* (BO) and *Analytical* (AN). In addition, the customer who interacts with the CRM system is also considered a role in the enterprise viewpoint because it represents an entity external to the CRM system. Business roles and the interactions among them are depicted in figure 6 and detailed as follows.

- *Customer*: performed by objects that represent the CRM system' customers, concerning policies related to customers location, current situation, preferences, etc. Customers may interact with objects performing CO role;
- CO: performed by objects representing the applications and devices, or groups of applications and devices that interact directly with the customers, such as IVR devices, human agent workstations, web connections, etc;
- FO: performed by objects which perform activities such as: customer care services, contact

management, telemarketing and sales force automation;

- BO: performed by objects responsible for the core activities of the business where the CRM system is applied to, concerning ERP systems, legacy systems, SCM systems and operational databases;
- AN: performed by objects responsible for the analytical CRM which perform activities that give intelligence to the CRM process, providing the other functional groups with policies so that the customer expectations could be better fulfilled. This role extends to data warehouses, data mining applications and OLAP (On-Line Analytical Processing) tools.

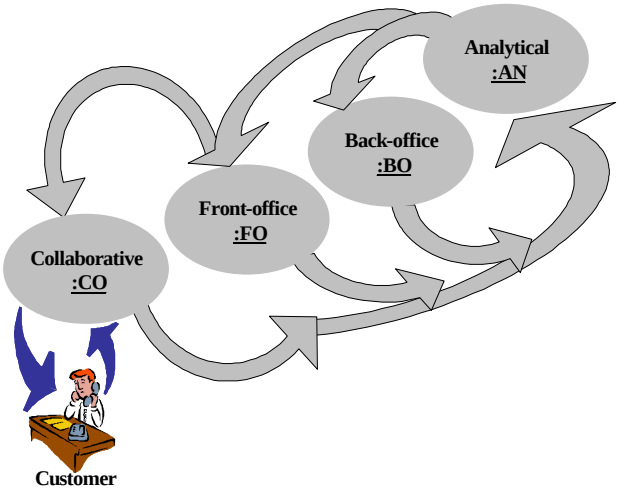

Figure 6. CRM communities in the enterprise viewpoint

The roles detailed above also define communities, each one containing sets of objects performing the same role. The interoperation among these communities is crucial to the one-to-one process and, consequently, to the CRM objectives.

So, all those communities must comprise a federation whose primary objective is to provide a one-to-one service to the customer. A service can be further modeled as a set of interactions among the CRM communities. For example, the analytical community (AN) may detect a business opportunity and request the telemarketing automation system (FO community) to contact some customers to offer a product employing a customer channel, such as: voice, e-mail or postal delivery (CO community). For the customers that accept the offer, the back-office community (BO) will process their orders. Anyway, the analytical system will process the customer responses in order to learn a little bit more about them for future contacts. So the CRM goals will be achieved.

3.3. Information viewpoint

In this work, the information viewpoint is used to model the general structure of information concerning the CRM system. This viewpoint may deal with the invariant, static and dynamic information schemas, but in the case of the generic architecture for CRM systems, the invariant schema is more relevant to provide a class diagram representing the structure of the information essential for a CRM process. Thus, figure 7 depicts such class diagram, which identifies the following information classes concerned with the CRM process:

- InfComponent: models information concerning the devices used in the communities CO and FO, where specifics characteristics of each of these communities are carried out by the specializations InfChannel and InfApp, detailed below.
- InfChannel: models information concerning the touch-points (or channels) with customers, such as: IVR (Interactive Voice Response) devices, telephonic branches, web-server connections, etc;
- InfApp: models information concerning front-office applications, such as help desk, sales force automation systems and market automation systems;
- InfExAgent: models external entities (probably a customer) in touch with the CRM system in a specific moment, such as during a phone call for the company's call center;
- InfInteraction: models associations involving external entities, customer channels and front-office applications, representing a well defined interaction of a customer with the CRM system;
- InfContact: models groups of inter-related interactions;
- InfCustomer: represents each customer, comprising every piece of information related to him or her;
- InfProduct: represents products and/or services offered to the customers;
- InfDeal: models the relationship among a customer, products (or services) and the contact (group of interactions) that drawn the customer to the product;
- InfCampaign: models campaigns about products, relating products to groups of customers likely to appreciate them, and involving appropriate front-office applications to offer these products to the customer.

In the context of a CRM system, the static and dynamic schemas are not so general as the invariant schema. The static schema defines specific states for each instance of the information classes identified in the invariant schema. For example, an instance of the class InfCanal, representing a telephone line, may have an attribute *status* that can be *idle*, *busy*, *disabled* and *fault*.

These values comprise states represented by the static schema.

Dynamic schemas can be used to define state transitions among the states identified in the static schema. For example, the behavior of a telephonic line can be described by a state diagram involving the states *idle*, *busy*, *disabled* and *fault*, as well as the events and conditions that trigger transitions from one state to another.

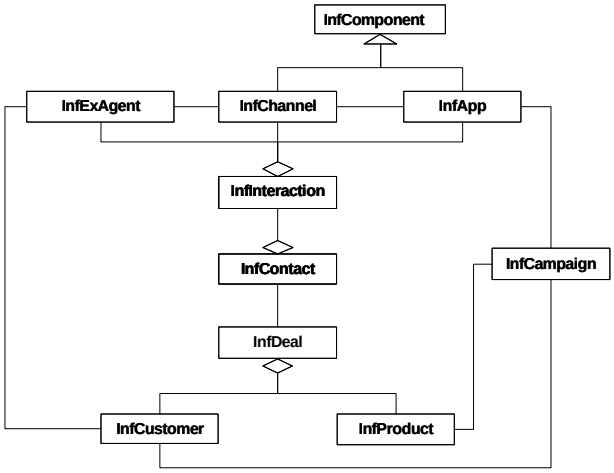

Figure 7. CRM static information schema

4 Middleware for integration among computer architectures

Enterprises applications have a large range of independent systems sometimes without interactions or fragile relationship [8]. On the other hand, dynamic advances in IT (Information Technology) increased complexity and customers demand for distributed information shown necessary a new way to manage IT systems.

4.1. Distributed processing problem

The relationship between different applications may be transparent, receiving and sending requests without new codes or additional application.

All companies have distributed processing applications without interactions allowing double effort. Today, middlewares (management, availability and basic communication layer) have functions to interact with applications available for these stand alone systems.

The companies' wish is to allow systems and databases integration within enterprises and across enterprises, collaborations, mergers, acquisitions and the Internet (a totally unstructured data source), solving problems concerning heterogeneity and distribution. It is not a wish, is a necessity around the computational world because the information is an asset more valuable than the company's facilities. By this way, accomplishing this whish is a complicate task and it is a challenge for developers and researchers [9].

In this context we are developing a multipurpose middleware, whose task is to allow integration and provide interoperability to distributed computer architectures. This middleware has been desiged to be used like an applications integrator providing interoperability between these applications. It is a necessary step for convergence between technologies.

RM-ODP brings a general architectural view for this middleware as well as global conceptual definition, analytical structure and standard specification. This section focuses on business information (information flows and structures, restrictions and standards) and computational viewpoint (it is a real need for the system).

4.2. Enterprise viewpoint

It outlines a middleware architecture that will integrate independent and different applications, being capable to access distributed data and execute distributed tasks. This middleware will provide information wherever, however and whenever the user request. Therefore, this middleware architecture will be a distributed programming model and will allow communication for all applications in several scenarios with flexible customization and configuration.

At the enterprise viewpoint, this architecture will install a component in existing applications that will intercept the requests and send them to the new middleware. When receiving the requests, this middleware will provide the best way available to attend it. This execution will be transparent to the users, *i.e.*, the requester and executor are not publicly visible. A given application will take part of a new system comprising several architectures. Therefore, this application will be an object for this new system with its functions and it will access and accept others applications' requests.

This architecture improves the use of a corporative system, assuring information integrity, quality of service (QoS) levels and availability.

4.3. Information viewpoint

By information view, the component installed in an application responsible for the interception of requests and redirection to the middleware is the request object.

Another component, also installed in the existing application, will receive the answer and will send it to the final user. It is a reception object. This mechanism is shown at figure 8.

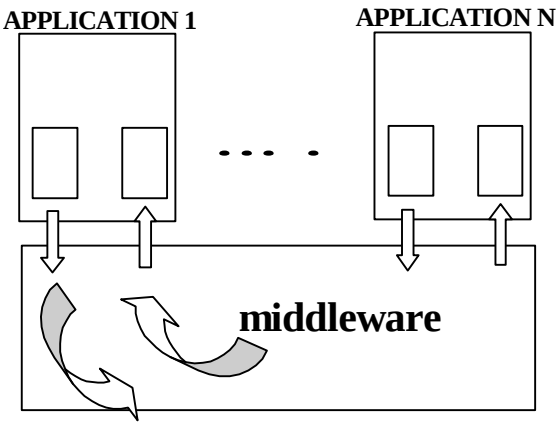

Figure 8. Information flow in the new architecture

The middleware receives, dispatches and sends the requests and answers to its users transparently. It is a way to accomplish heterogeneous management.

An existing architecture connected to this middleware will receive and send requests, integrating independent systems with different architectures without different gateways or bridges between each system.

5 Conclusion

The different approaches presented in this article show how RM-ODP can be properly used for the specification and modeling of distributed systems targeting different problems, which restates how RM-ODP may be malleable for architecting of distributed systems.

In the three cases presented, the studies show a weak points in RM-ODP when it was used for modeling systems where one or more blocks are legacy systems. Furthermore, for the implemetation the concern is the lack of compatibility between RM-ODP and distributed objects architectures like CORBA and J2EE.

Despite de fact that RM-ODP is not proper to formally specify an entire system, which would require to dig into its implementation details – in fact, such deed would be at least impracticable –, RM-ODP is appropriate to determine the architectural patterns necessary to drive the further development of the system, once RM-ODP modeling requires a good understanding of the interactions of the system with its environment, the elements that comprise the system, their interrelationships and the activities that must be performed, which compels the architects and designers to appropriately reason what to specify. Once a good architectural specification is ready, it is possible to provide an implementation for it using well-grounded technologies, basing on the ODP technology viewpoint.

6 References

[1] Presidencia Española de la Unión Europea, *El Potencial de la Convergencia Tecnológica en el Desarrollo de la Sociedad de la Información*, Colegio Oficial ingenieiros de telecomunicación., 2002.

[2] H. Balen, "Distributed Object Architectures with CORBA", *Sigs Books*, Cambridge University Press, 2000.

[3] G. Blair, G. Coulson, N. Davies, "Standards and Platforms for Open Distributed Processing", *Eletronics & Communication Engineering Journal*, p.123–133, 1996.

[4] ISO/IEC 10746-1, *Information technology – Open Distributed Processing – Reference model: Overview*, 1st edition, 1998.

[5] E. Shahnam, "The Customer Relationship Management Ecosystem", *Delta Research Reports*, 2000.

[6] A. P. Gonçalves, "Proposta de Arquitetura Aberta de Central de Atendimento", Master's dissertation, *EPUSP*, 2001.

[7] J. D. Wells, W. L. Fuerst, J. Choobineh, "Managing Information Technology for One-to-one Customer Interaction", *Information & Management*, 1998.

[8] M. Feridum, G. D. Rodosek, "Management of IT Services", *Computer Networks*, v.43, pp 1-2, 2003.

[9] K. Geihs, "Middleware Challenges Ahead", *Computer*, pp 24-31, June 2001.

Acknowledegements

The authors wish to thank the support of the UNILINS - Escola de Engenharia de Lins - Lins, Brazil.