Multi Channel Architecture Model Based on Service Oriented Integration

Ion LUNGU, Davian POPESCU, Anda VELICANU
Academy of Economic Studies, Bucharest, România

The volume of data and numerous applications developed within a company can often generate a redundancy difficult to control. In the same time, the homogeneous or heterogeneous management systems of the companies become overcharged for obtaining useful information from databases. For this reason, the organizations develop specialized systems for the integration of existing applications and data. To achieve these systems, there are used a number of technologies, methods and architectures such as SOA architecture. In this article, are presented the components of SOA architecture, its advantages and a solution for integrating applications at the Presentation Tier.
Keywords: *application integration, SOA architecture, integration model, architectural levels, information systems.*

1 Introduction

The information system of an organization is a constantly increasing entity. Yet there is a risk that the new requirements, the new business requirements determine the adoption of new specialized systems with the immediate consequence of generating a redundancy in business logic, data and responsibilities.
This scenario belongs to a not so far historical period in which the technological support did not offer solutions tailored to rationalize the applications' resources in the company. It is enough to remember the period in which there was achieved a vertical application for each subsystem (channel) used. Later, there came the need to integrate these applications through a point-to-point connection, using from time to time the specific integration way, best suited for that type of interaction. In this scenario, the proliferation of the systems and connections leads to a complexity represented by numerous interconnected entities (literature defines this as "spaghetti architecture") as can be seen in Figure 1.

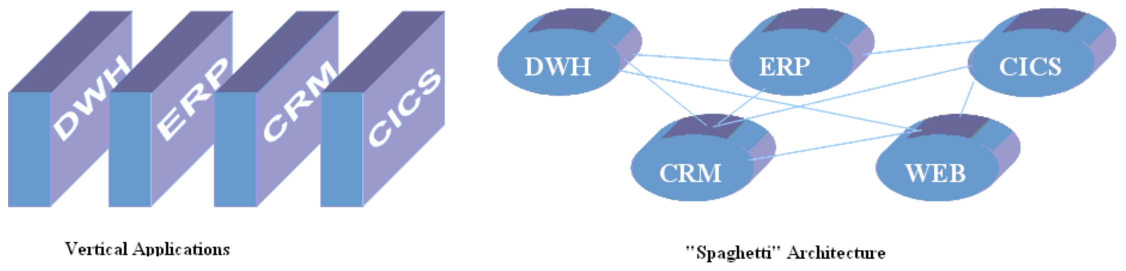

Fig.1. Integrating the vertical applications

Nowadays many organizations are still facing with the complexity of managing induced by the two approaches described in Figure 1. This management has obvious repercussions on management cost, and worse, very often it limits the development park of applications, where such developments require ad-hoc implementation and management of new channels for interaction with existing features.

2. The SOA Architecture Model – Solution For Integrating The Applications

At present there is a technological solution that offers alternative models and standard guaranteeing reuse, rationalization and so, efficiency. *The solution to the problems of rationalization is a SOA (Service Oriented Architecture) model, namely the adoption of a service-oriented architecture and integration* [5]. Business functions are provided by specialized systems and set to use through a standardized interface. Central Processing Unit becomes "the service" seen as a reusable software used through a request / reply interaction, which automatically runs a business function (Figure 2):

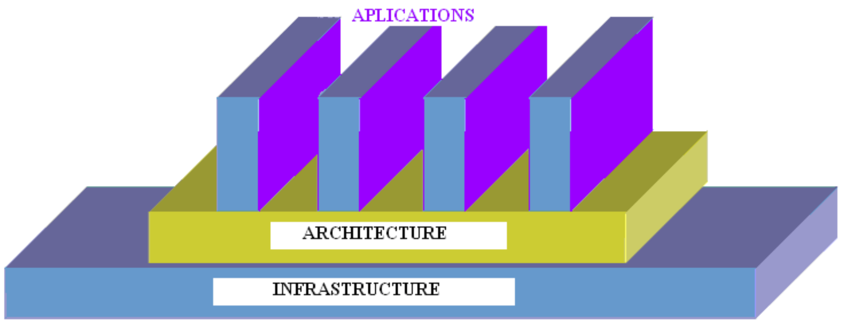

Fig.2. The new service-oriented strategy

The services are modular elements that can be composed as needed; the architecture ensures the technology independent interoperability in which these services were performed or natively displayed for use.
The main advantage of SOA model adoption lies in drastically reducing the complexity of the general infrastructure because the applications' logic is organized in specific modules or services, so it's not duplicated in several systems. Also the number of the managed connections is the same as the number of co-opted platforms.
Another fundamental strategic advantage is the fact that, using a SOA architecture can easily co-opt information systems, resulting integrated processes belonging to external entities, such as partners or suppliers. The purpose is to automate these processes, as much as possible, and to hasten the return information to and from the outside.
Even if many of these business processes are created for their internal use within the company, their evolution leads to sharing this logic between the companies or foreign partners. Using services exposure infrastructure, not only it integrates applications with the aim of sharing information, but it provides the infrastructure for reusing the business logic in inter-company processes (by using a business process oriented approach) [2].
In recent years, there has been brought a big help in this area due to the introduction of Web Services standard. This is a today de-facto standard for interoperability of platforms that allows the use of Internet penetration to provide access to remote applicative services.
Another advantage of using SOA architecture is that streamlining the services provides, to those who are dealing with businesses within an organization, a more concrete vision of the functions park, data and responsibilities in the company's information system. The introduction of SOA architectures are not an impact on existing systems. It may take a gradual approach of short steps through establishing and continuously growing a service's park published and used according to the priorities of the users [3].
A good approach, in terms of design software, considers that the best solution is to structure the development of applications in layers / tiers, defining the environment and the roles of each. In this way, specific components can be isolated and may eventually independently evolve from the rest of the structure. Benefits can be identified in simplifying the maintenance of a system, in which there is no logic duplication, but also in the effectiveness of components which, being specialized, are optimized to perform their tasks.
The philosophy that drives the design of this architecture is the "separation of logics". The multichannel architecture and the service-oriented architecture should enable the development of applications in which there are differentiated logics that depend on a specific channel used for cross-linkers, business, and which implements a business' processes. The

architecture imposes some rules for developing the applications, scheduling them, and providing services that allow abstracting the development applications of any technological connotation, with their simplification as result. These services are provided by forcing the positioning of certain logics in places well defined.

In the light of multichannel and service orientation, the logical architectural model identifies the tiers shown in Figure 3. Within each tier there are included architectural services (shown in the figure) and applied components [1].

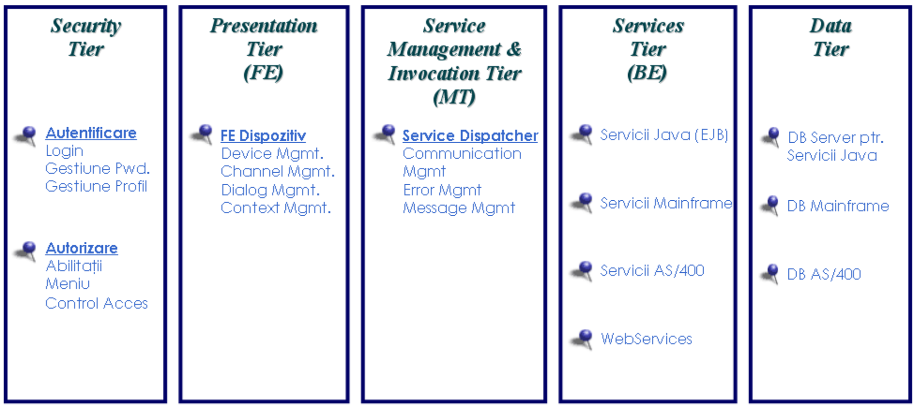

Fig.2. N-tier model: positioning the components

Security Tier - is the architectural tier which centralizes all the operations relating to security, both to identify the user and his right to access a system (authentication) and to control over its operability by defining areas in which the user is allowed to operate (authorization), in terms of business functionality and of data visibility.

Presentation Tier - Front End - is dedicated to the management of the interaction between various users and the system, transforming the user's requests in invocations of the business procedures through the management level, and organizing and formatting the output with the channel and the used device in a consistent way, not to mention the practical user profile. At this level there are managed the logics that govern the conversational flow of the various functions and the logical presentation of the system.

Service Management & Invocation Tier - Middle Tier - is the architectural level that separates the presentation tier from the business logic, both in terms of functionality and technology. This tier is interposed between the interface, which manages the dialogue, and the interaction with the user and the services tier, where the functional applicative logic can be found. At this level there are contained components that allow the integration of heterogeneous technologies and platforms, eliminating the issues of integration of each application.

Services Tier - Back End - is the functional nucleus of the system, where can be completely found the applicative logic that accesses the business data. In a service-oriented approach, each service is independently developed, that is, any implementation is independent of the previous executions. How it works depends exclusively on the entry variables and not on the previous history. The service tier makes it possible to expose the various applications' services through published interfaces that will be raised through the Middle Tier, by the various applicative dialogues.

Data Tier - is the logic tier in which can be found the business data. Developing the software provides that the access to data will not be attached to every practical way, but will be centralized in specific data access modules (Input / Output modules) in which attention is placed on the physical database, exposing to applicative services logical structures that are independent of the specifics of one or more DBMS (Data Base Management System).

3. A Solution For Integrating An Application At The Presentation Tier

The presentation tier can be designed using the MVC (Model-View-Controller) paradigm, which is considered by far the most appropriate. It became a de-facto standard (practically unique) for web software architectures. MVC emerged as a methodology to separate and detach functionality of development applications and it is most suitable for exposing the logic in a multichannel service-oriented architecture.

The paradigm requires thinking an application in the following logic tiers:

o the model - the data or service (if SOA);

o the view - is the way to view the data or the results of a SOA service;

o the controller - manages each user's request, supervising the user's browsing and providing services to access business logic and components for viewing data returned to the user.

Two of the immediate benefits offered by the use of MVC are the net separation of the logic view from the business logic and the existence of a unique point of entry into the system where it is thus possible to centralize several "technical" behaviors of an application. A controller that can identify the device/channel or the language and the location of the user in order to compel the architecture to different behaviors depending on the user receives all requests sent to an application. The controller has the ability to perform checks on the correct user's navigation and it can make requests to the security tier's components to check the user's ability regarding a specific feature /a business request.

Presentation Tier's components, listed below, will be carried out as extensions of the Struts framework, the most widespread implementation of the MVC paradigm. So it is used as the kernel of the Front-End architecture. Using Struts as a frame of reference is a technological choice, which is a de-facto standard in J2EE.

The need to develop applications in SOA architecture (multichannel / multi-device, multi-company) requires improving the baseline with other specific components. The diagram in Figure 4 illustrates the components and their interactions in order to achieve a Front-End architecture. In blue there are drawn the architectural components, namely those parts which do not depend on a specific application. With orange, there are listed applicative units or those software units representing the various applications hosted by the architecture.

Further, there are described the features of all the components in order to give a panorama of how complex the presentation tier operates. The complexity of the architectural scheme can be reduced in the development of future applications.

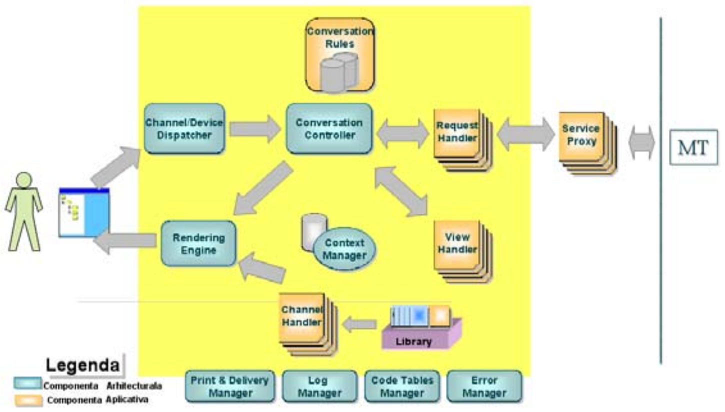

Fig.3. Schema of the Presentation Tier's components

In terms of architecture that we have proposed, there can be identified the following entities for an application:

o the application - a set of business processes with the same logic;

o the process - a whole consisting of the user

interface and the logic that implements a business process (for example: the search process for selecting an action with the aim of its trading);
o the function - each component function of the process;
o the dialogue - is a part of a function (the implementation) in terms of user's interface and related functions of a business (for example: selecting a stock symbol). Any function is made up of at least one dialogue.
In Figure 5 are presented the inclusion / structure relations between the entities defined above:

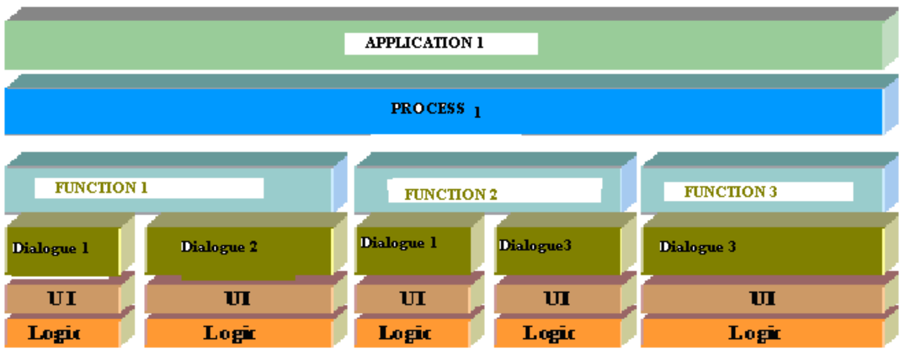

Fig.4. Entities identified in the analyzed architectural model

The composition of dialogues in functions is made using the following mechanisms:
1.GOTO: within Function 2, at the end of Dialogue 1, is running Dialogue 2;
2.CALL: a dialogue can not call another at its end but at a random time of its life cycle and it suspends the implementation (suspension phase); at the end of executing the called dialogue, the initial dialogue resumes the flow of navigation in the exact same point where it was discontinued (resumption of execution phase).
From this solution there can be inferred also the possibility of re-using some dialogues in several functions (such as Dialogue 1 is reused in Function 2).

4. Conclusions

The realization of each service in part is not only depending on the execution technology platform. It also depends on the functional requirements or the need to re-use components and programs that already exist. Usually the implementing the services should follow a paradigm on three tiers, which separates into distinct modules or components:
o logical access to data;
o elementary business logic that implements the minimum re-usable features;
o logical complex that aggregates and develops individual elementary logic in order to provide a service with a business signification well defined.
It is obvious that this definition is qualitative and open to various interpretations, and the design of a service can not ignore a phase of analysis of functional requirements and eventually of the individual sub-elementary or already completed logic. In general, the recommendation is to maintain a strict separation between the access to data and the decomposing in elementary consistent logic, having the characteristics of a business and the number of potential reuse.

Refereces:

[1] David S. Linthicum – „Next Generation Application Integration: From Simple Information to Web Services", Addison-Wesley Pub Co; 1st edition, 2003
[2] Dirk Krafzig, Karl Banke, Dirk Slama – „Enterprise SOA : Service-Oriented Architecture Best Practices (Coad)", Prentice Hall PTR; 1st edition, 2004
[3] Douglas K. Barry – „Web Services and Service-Oriented Architectures: The Savvy Manager's Guide", Morgan Kaufmann, 2003
[4] James McGovern, Scott W. Ambler, Michael E. Stevens, James Linn, Elias K. Jo, Vikas Sharan – „The Practical Guide to Enterprise Architecture", Prentice Hall PTR; 1st edition, 2003
[5] Lungu I, Bologa R, Diaconiţa V, Bâra A, Botha I – „Integrarea sistemelor informatice", Editura ASE, 2007
[6] Thomas Erl – „Service-Oriented Architecture : A Field Guide to Integrating XML and Web Services", Prentice Hall PTR, 2004