SUMMARY

There is a considerable difference between academic perceptions of Software Architecture and industrial practice ... It is interesting to note that sometimes the problems that the industry identified as the most important and difficult, not identified or considered not - problems at the academy.

Software architecture is to have a global and abstract view of system it wants to develop, allowing us to make the right decisions, set the parameters and boundaries of the system, taking into account the functional and nonfunctional requirements.

Generally, it is necessary to invent a new software architecture for each information system. Typically known to adopt an architecture based on its advantages and disadvantages for each particular case. Among the major architectural patterns are: layers (Layered Systems), Pipes and Filters (Pipe and Filter), Blackboard (Blackboard) Broker (service-oriented architecture) and Model View Controller (Model View Controller).

For managing business data technology platforms provide the means to collect data from data providers, transport, storage and processing. In addition to delivering them to consumers. Among the most widely used technology architectures have JEE (Java Enterprise Edition), Microsoft. NET and PHP.

Using the methodology of software architect in the development of our applications software quality achieved with the following parameters: efficiency, reliability, usability, maintainability, expandability, interoperability, re usability, integrity and portability.

INTRODUCTION

In the early days of computing, programming is considered an art and developed as such, because of the difficulty for most people, but over time has been discovering and developing forms and general guidelines, based on which can solve the problems.

At this, they've called Software Architecture, because, like the plans of a building or construction, they indicate the structure, functioning and interaction between parts of software.

In his book *"An Introduction to Software Architecture"* David Garlan and Mary Shaw define that *"Architecture is a design level that focuses on aspects beyond algorithms and data structures of computing, the design and specification of the structure is a system-wide problems."*

We use the term *"Architecture"* versus *"Design"*, evoking notions of codification, abstraction, formal training standards (for software architects) and style. It is time to re - examine the role of software architecture in the broader context of software process and its administration, and to identify new techniques have been adopted.

Then there is adequate clearance to begin to understand the entire software development process in the context of current trends in theory and practice of architecture. What will help us to correlate the basic research and academic contributions to the visions and requirements of the industry.

TABLE OF CONTENTS

1 SOFTWARE ARCHITECTURE

Software Architecture is to have a global and abstract view of system it wants to develop, allowing us to make the right decisions, set the parameters and boundaries of the system, taking into account the functional and nonfunctional requirements.

Software Architecture comprises a number of decisions about the organization of a system: the selection of structural elements and interfaces by which a system is made, along with his behavior that is specified by the collaboration between these elements.

1.1 SOFTWARE ARCHITECT

Software Architect term has become fashionable in the title of company-wide systems or systems own area. Fashion say, because not all companies really need software architects, and perhaps even all projects need a true software architect.

It is common for many tasks relevant to a project can be resolved with an experienced developer, without the need to hire an architect. Too often we tend to confuse these two profiles, which are abysmally different. Also note the difference between “technological gurus” and the real architects.

1.2 ICEBERG OF USABILITY

Until recently, it was assumed that usability was an exclusive presentation of

information. It was thought that presentation encapsulating layer separated from the rest, you could develop the application, and iterative usability tests to pass. After each test, so you only need to solve the problems by modifying the presentation and because of this separation, the feature will not be affected.

Usability and Software Architecture: It takes into account the usability since the beginning of software development project, ie what is called time software architect. We also know that later when problems are detected more difficult to fix, how many times has happened that when we are designing the interface of a new system we want to create dialogue and interactions that the technological environment does not allow us, and we ended up crying but if this I have done in another application!. If we analyze these scenarios of interaction, we see that the cause of which cannot be implemented is that no consideration was given to the user at the beginning of system design, ie the software architecture.

1.3 ARCHITECTURE DESCRIPTION LANGUAGES

An Architectural Description Language (ADL) is a descriptive language modeling that focuses on high-level structure of the application rather than the implementation details of individual modules. An ADL is a language that provides features for modeling the conceptual architecture of a software system, distinct from the implementation of the system.

Criteria ADL Definition: The criteria we consider the are those found in Vestal, who argues that an ADL should model or support the following concepts: components, connections, hierarchical composition, computing paradigms, communication paradigms, sub adjacent formal models, tool support, and automatic code composition applied.

Languages: None of the languages were imposed or in academia or in the market, a bar with new ADL's that are in the market such as UML 2.0, domain-specific languages (Domain Specific Languages), XML and Semantic Web.

1.4 ARCHITECTURAL VIEWS

The architectural view represents a partial aspect of a software architecture that shows specific properties of the systems. The system architecture consists of multiple views associated with different dimensions or perspectives of the system, no particular view is the system architecture.

Architectural view of John Zachman: It consists of an array of six rows and six columns, consisting of 36 cells. Has six levels of architecture developed in row: scope, business model, system model, technology model, details of representation and functional systems. The first three levels are conceptual and the following are details of design and construction of systems. In columns represent different areas of interest: data, processes, networks, people, time and motivation. They correspond to the what, how, where, who, when and why.

Architectural View Philippe Kruchten: Philippe Kruchten proposed the model “4 +1”, linked to the Rational Unified Process (RUP), which defines four different views of software architecture: logical view, process view, physical view and development view. A fifth view is a selection of use cases or scenarios that architects can develop through the earlier hearings.

Architectural View of Grady Booch, James Rumbaugh and Ivar Jacobson: It consists of an outline of five views which show different information about the

system. The architectural view proposed by Grady Booch, James Rumbaugh and Ivar Jacobson in his introduction to UML where you are: use case view, design view, interaction view, view of implementation and deployment view.

Architectural view of Bass, Clements and Kazman: Bass, Clements and Kazman presented in 1998 a model that consists of nine views, oriented towards the specific design and implementation: module structure, logical or conceptual structure, physical structure, structure use known structure, data flow, control flow and class structure.

1.5 ARCHITECTURAL STYLES

"Architectural Styles are a set of design rules that identify classes of components and connectors that can be used to compose the system or subsystem, together with local and global restrictions on how the composition is carried out."

Among the architectural styles are:

Data-centric architecture: At the core of this architecture is a data store (eg a document or database) to access other components often to update, add or delete data from storage, between repositories this are: a passive repository and active repository (blackboard). Data Centric Architectures provide integrity, there may be changes in the components without affecting the other *Client Software*, together with mechanisms for transferring information between the *Client Software*.

Data Flow Architecture: Architecture applied when the input data are

processed through a series of computer components in the output data. This pattern is composed of a set of components called “filters” connected by “pipes” that carry data from one component to another. The filters are designed to receive input data in a form and produce output data in a specific way. If the data flow will produce a line of transformations is called *Sequential Batch*.

Call and Return Architecture: Software Architect provides the structures of simple exchange programs and scalability. Style used in large scale systems, in this style we have two sub-styles such as main program / subroutine, remote procedure call.

Object-Oriented Architecture: Architecture in which system components encapsulate data and operations that are used for handling them. Communication and coordination between components is done by sending messages.

Service-Oriented Architecture: Service Oriented Architecture or SOA design provides a framework for application integration independent so that the network can be accessed from their features, which are offered as services. How to implement it using Web Services, a standards-based technology and platform independent, with which SOA can break down monolithic applications into a set of services and implement this functionality in a modular way.

Aspect-Oriented Architecture: This architecture helps to improve the software development process, using the concept of appearance over the entire life cycle. Provides techniques for early identification of issues, extraction, representation and composition later. In this paradigm, the issues are considered first class entities that can be manipulated throughout the development process of a system. The definition of aspects is an abstraction

mechanism by which they can be incorporated into an existing system so that the items added are not scattered across different modules, but remain in separate modules.

Laminated Architecture: This architecture is structured in layers, each of which operates increasingly progressive approach to machine language.

2 PATTERNS OF SOFTWARE ARCHITECTURE

2.1 DEFINITIONS

Architectural patterns are considered as templates for specific software architectures, which determine the structural properties of an application and have an impact on the architecture of subsystems.

2.2 LAYERS ARCHITECTURE

The Layers architectural pattern helps to structure applications that can be broken down into groups of sub tasks in which each group of sub tasks is at a particular level of abstraction.

Structure: There are several ways to implement the philosophy of layers, depending on which of the following scenarios are expected to fulfill:

1. Each layer N 1 sends service requests to objects on the bottom layer N, which in turn is based on requests to lower layer. Typically there is a

cascade of requests, ie to satisfy a request to a layer N 1.

2. Each layer N notifies its upper layer N +1 that has occurred an event of interest. N +1 layer can gather several events before notifying his superior in turn.

3. As (1), requests are handled in a top - down, but they do not need to reach the innermost layer or layer 1.

4. As (2) but the notifications did not reach the top.

5. N involves two cell layers that communicate with each other.

2.3 PIPES AND FILTERS ARCHITECTURE

The pattern of pipes and filters architecture provides a framework for system data processing flow. Each process step is encapsulated in a filter component. The data is passed through pipes between adjacent filters. Recombining filters allows you to build families of related systems.

Structure: The pattern of pipes and filters architecture consists of:

- **Tube:** It's the connector that passes data to the next filter. This is a unidirectional flow of data, usually done by a data buffer to store all data, until the next filter has time to process.
- **Filter:** It is responsible for filtering or transform the data it receives through the pipes.

- **Pump (input):** Also known producer is the data source. Can be files, databases, or input devices.

- **Sink (output):** Also known as a consumer is the target data can be files, databases or output devices.

2.4 BLACKBOARD ARCHITECTURE

The Blackboard architectural pattern is useful for problems for which no known deterministic solution strategies. Blackboard Architecture in a number of specialized subsystems can pool their knowledge to build a possible partial or approximate solution.

Structure: A blackboard system often has the following structure:

- **Source of knowledge:** It is a component that is added to the solution of the problem. It can be anything that is read from a board level, and proposes a change to parts of the board. Its most common form is a production rule. A source of knowledge is completely alien to other sources of knowledge.

- **Blackboard:** Data structure is common knowledge sources. The board is capable of representing all the states of a problem space. The board has several levels of description with respect to the problem space. These levels may have several relationships with others. The levels are parts of the same data structure. If you need to separate data structure, the board is divided into panels. Each panel in turn can contain multiple levels.

- **Shell control:** Determines which source of knowledge has the opportunity to change the board. Each execution cycle, identifies changes to the board enable appropriate knowledge sources, select one of these and running.

2.5 BROKER ARCHITECTURE

The Broker architectural pattern can be used to structure distributed software systems with decoupled components that interact by remote service calls. A broker component is responsible for coordinating communications, such as forwarding requests, as well as for transmitting results and exceptions.

Structure: The Broker pattern consists of six components involved:

- **Client:** These are applications that access the server services. To invoke remote services, clients send requests to the broker. After the transaction was executed, customers receive replies or exceptions to the broker.

- **Server:** Implements objects expose their functionality through interfaces which consist of operations and attributes. The interface is available through an *Interface Definition Language* (IDL) or a binary standard.

- **Broker:** A messenger responsible for transmitting requests from clients to servers, and the transmission of responses and exceptions from servers to clients.

- **Bridges:** These are optional components used to hide the

implementation details when two brokers inter operate. Suppose that a Broker is running in a heterogeneous network. If requests are transmitted over the network, different brokers should be reported regardless of the network and operating systems used.

- **A Proxy client:** They represent an additional layer between clients and the broker, to provide a transparency in the sense that a remote object appears to be local to the client, ie, hides the implementation details.

- **A Proxy server:** They are generally similar to the client-side proxies, the difference is that they are responsible for receiving applications, unpack incoming messages, the unmarshaling parameters, call the appropriate service, and the marshaling of results and exceptions before sending to the client.

2.6 MODEL VIEW CONTROLLER ARCHITECTURE

The design pattern Model View Controller (MVC) divides an interactive application into three components. The model contains the basic functionality and data. The view shows the information to the user. The controller handles user input. The view and controller together constitute the user interface. A change in the propagation mechanism ensures consistency between user interface and model.

Structure: The basic structure of this architecture pattern is as follows:

- **Model:** The object representing the program data. Manages data and controls all its transformations. The model has no specific knowledge of

the controllers or views, does it contain references to them.

- **View:** The object that handles the display (GUI) of the data represented by the model. Generates a visual representation of the model and displays the data to the user. Interacts with the model through a reference to the Model itself.

- **Controller:** It is the object that provides meaning to user commands, acting on the data represented by the model. When a change is made, goes into action, either by changes in the information model or changes in View. Interacts with the model through a reference to the Model itself.

3 TECHNOLOGY ARCHITECTURE

3.1 JEE (JAVA ENTERPRISE EDITION)

According to the official website of Sun Microsystems: "Java EE is a technology architecture that is based on the solid foundation of Java Platform, Standard Edition (J2SE) and is the industry standard for enterprise application development, service-oriented applications (SOA) and next-generation web applications. "

Enterprise Java Beans: A standard distributed component model server-side Java Platform Enterprise Edition (JEE). EJB technology enables rapid and simplified development of distributed applications, transactional, secure and portable Java technology-based. The EJB's are one of the API's that are part of standard business application building J2EE (now Java EE 6) of Oracle

Corporation.

The objective of EJB's is to provide a model that allows developers to abstract from the general problems of business applications (concurrency, transactions, persistence, security, etc) to focus on developing business logic. There are three types of EJB's:

- **Entity EJB's:** Encapsulate the server-side objects that store data. Entity EJBs have the fundamental characteristic of the persistence, container managed persistence and bean-managed persistence.

- **Session EJB's:** Manage the flow of information on the server, serve customers and access to services provided by the other components are the server, with a session (stateful) and non-session (stateless).

- **Message – Driven EJB's:** Beans with asynchronous operation. Using the Java Messaging System (JMS), they subscribe to a topic or queue and are activated upon receiving a message addressed to the topic or queue.

Java Server Faces: A technology for Web-based Java applications simplifies the development of user interfaces for applications in the Java platform Enterprise Edition (JEE). JSP technology uses JSF (Java Server Pages) to make the deployment of the pages.

It has now spread to integrate JSF with Ajax rich clients such as:

- **RichFaces:** It is a component library for JSF and an advanced

framework for easily integrating AJAX capabilities into business applications.

- **IceFaces:** Es more than a library of JSF components and AJAX is an AJAX framework JEE to develop and deploy rich business applications, like RichFaces; ICEfaces comprises similar advantages.

JDBC (Java DataBase Connectivity): It is the industry standard for database connectivity independently of the programming language Java and a wide range of databases - SQL databases and other tabular data sources such as spreadsheets or flat files. The JDBC API provides a call-level SQL-based access database.

JDBC technology allows you to use the Java programming language to exploit *"Write Once, Run Anywhere"* capabilities for applications that require access to company data. JDBC technology-enabled and your controller, you can connect data from the company, even in a heterogeneous environment.

3.2 .NET (MICROSOFT)

". NET framework is the platform of managed code development for Microsoft. This consists of a set of tools and libraries that can create all kinds of applications, from traditional desktop applications (Windows Forms or WPF) to applications for Xbox (XNA) through web development (ASP.NET), development Mobile (compact framework), server applications (WPF, WCF), and so on. "

The main components of Microsoft. NET are:

- **Common Execution Environment (CRL):** .NET provides a runtime environment called Common Execution Environment, which runs the code and provides services that facilitate the development process.

- **Base Class Libraries (BCL):** .NET includes classes, interfaces and value types that accelerate and streamline the development process and provide access to the functionality of the system. For this they use the CLS (Common Language Specification), therefore can be used across all languages .NET.

3.3 PHP

"PHP is a general-purpose scripting language widely used, especially designed for Web development and can be embedded into HTML."

The current version of PHP 5 includes the following improvements: better support for programming reorient PDO object, performance improvements, better support for MySql completely rewritten extension, better support for XML (XPath, DOM, etc), native support for SQLite, integrated support for SOAP, data iterators, exception handling, and improvements to the implementation of Oracle.

PEAR: According to the official website of the PHP PEAR (PHP Extension and Application Repository) is a development environment and distribution system for PHP components. "

Frameworks: The most important frameworks that has PHP are:

- **Zend Framework:** No installation, just need PHP 5 and incorporates the standard MVC (Model View Controller).

- **Symfony:** It was designed with the goal of creating web applications with the use of its features. It has a library of classes that can reduce development time. Symfony is developed in PHP 5, requires installation, configuration and command line includes the MVC (Model View Controller), supports AJAX, templates and a large number of database engines.

4 APPLICATION

4.1 PROJECT MANAGEMENT

The project should provide an answer to the design of the prototype of a System of Enterprise Resource Planning - Finansoft. Which consists of the following modules:

Contabilidad: Plan de Cuentas, Jornalización, Mayorización, Balances y Estados Financieros.
Facturación: Clientes y Ventas.
Inventario: Productos, Proveedores y Compras.
Recursos Humanos: Inventario de Personal.
Seguridad: Usuarios, Roles y Menús.

Assumptions and restrictions on the system and that are derived directly from interviews with the stakeholders of the company are:

a) Should be considered the implications of the following critical points:
   a.a) Insurance systems, data protection, security of data transmissions, etc..
   a.b) Workflow management, security of transactions and information exchange.

b) The automation of the internal management of the register must comply with current legislation.

c) The billing module should be developed as an independent in desktop application for use by all branches of the company.

It should be noted that according to the philosophy of RUP (and all iterative and incremental process), all artifacts are objects of changes during the development process, thus, only the end of the process could have a final version and complete each of them.

Project Participants: Staff involved in the project consists of the following jobs and associated personnel:

- ***Project Manager:*** Ing. Irving Reascos.
- ***Software Architect:*** Egdo. Alcides Rivera Posso.
- ***Software Enginner:*** Egdo. Alcides Rivera Posso.
- ***Programmer:*** Egdo. Alcides Rivera Posso.

Users need to have a web browser only. Reports can be generated both in pdf,

html, word processors and spreadsheets.

To ensure the completion of a reservation transaction intended to minimize weight and using graphics to prevent the flexible use of the page. All terms of successful or unsuccessful transaction will be reported immediately to the user.

Software Features: The software features are proposed by the stakeholder of the company are:

- **CSW1: CONTABILIDAD:** The accounting department will have access to the entire module *Contabilidad.*

- **CSW2: FACTURACIÓN:** The sales department will have access to the module *Facturación* that takes care of sales and manage customer data.

- **CSW3: INVENTARIO:** The logistics department manages and operates the company's central warehouse, which is the main supply from other warehouses. Module available *Inventario* that will automate the process of replenishment of stocks of stores and refueling of the various warehouses.

- **CSW4: RECURSOS HUMANOS:** The human resources department is responsible for personnel management.

- **CSW5: SEGURIDAD:** The system administrator will be responsible for the management module *Seguridad* and administration of the servers where most of the system.

Stackholders: The user representatives and spokesmen for the company's

needs are the stakeholders. This project has only been treated with a stakeholder as a representative of users and business needs, but are divided representatively.

Actors: Defined this requirement to list the users of the system, this project has defined the following parties: *Contador*, *Jefe de Almacén, Vendedor, Jefe de Logística, Jefe de Recursos Humanos y Administrador.*

Objectives: The main objectives of the System of Enterprise Resource Planning - Finansoft is the automation of all processes for better control of all the products the store has also updated reports may be obtained from all processes. The main construction design and implementation has been that the application must run under a platform that consists of the following components:

1. **Programming Language:** Java.
2. **Development Enviroment:** NetBeans 6.9
3. **Server:** SuSE Linux Enterprise Server.
4. **Relation DataBase Management System:** Postgresql 8.4.3
5. **Application Server:** GlassFish 3.1
6. **Technology Architecture:** JEE (Java Edición Empresarial) 6.

Each of the components of the business is further divided into the three layers of architectural pattern Model View Controller (MVC): presentation logic, business logic and integration logic. Architectural styles that were used for the development of the prototype are: Object Oriented Architecture and Aspect-Oriented Architecture.