# DOCUMENTACIÓN Y ANÁLISIS DE LOS PRINCIPALES FRAMEWORKS DE ARQUITECTURA DE SOFTWARE EN APLICACIONES EMPRESARIALES

Presentado por:

**Ing. Hugo Fernando Sarasty España**

Director:
**Dr. Gustavo Rossi**

**Trabajo Final Integrador presentado para obtener el grado de Especialista en Ingeniería de Software**

**Facultad de informática – Universidad Nacional de La Plata**

**Octubre, 2015**

# INDICE

## INDICE DE GRÁFICAS

## INDICE DE TABLAS

# INTRODUCCIÓN

En la actualidad el avance de la tecnología y el dinamismo en los negocios, a determinado que las empresas requieran tener a la mano servicios tecnológicos que les permita facilitar y optimizar los procesos que diariamente llevan a cabo.

En este sentido el surgimiento de nuevas herramientas vienen a hacer el cambio en el entorno empresarial, la aparición y diseño de nuevas metodología no es un caso fortuito, ha sido el constante trabajo en donde se han visto envueltos, desarrolladores, diseñadores, ingenieros de software y arquitectos junto a los directores, gerentes y demás colaboradores de las empresas.

La necesidad imperante del manejo de la información ha permitido y de alguna forma ha inducido el desarrollo de nuevas tecnologías que sean aplicables y adaptables a entornos laborales. Estas nuevas tecnologías estan enfocadas a acompañar y optimizar todos los procesos que contiene una empresa.

El siguiente trabajo se enfoca en un tema común hoy en día en el ambiente tecnológico y empresarial, el cual es la arquitectura de software y su aplicabilidad a través de frameworks a proyectos empresariales. Este trabajo de investigación servirá de base para obtener un conocimiento y entendimiento de los frameworks de arquitectura de software más usados en el desarrollo de aplicaciones empresariales, determinando su aplicabilidad según el proyecto que se esté abordando.

## PROPOSITO

La arquitectura de software es una pieza fundamental dentro del desarrollo de sistemas de software estructurados, eficientes y escalables. Son muchos los sectores en los cuales el uso de software es fundamental, entre estos está el sector empresarial. Actualmente el sector empresarial es uno de los sectores con mayor consumo de herramientas sistematizadas, esto debido a que existe una relación directa entre la capacidad y éxito de los negocios de la empresa con el uso de software especializado, entre los cuales están: ERPs (Enterprise Resource Planning), CRM (Customer Relationship Management), software de facturación y software de gestión general. Teniendo en cuenta la importancia que tiene el uso de herramientas sistematizadas para el sector empresarial, es fundamental que los ingenieros de software o arquitectos de software, tengan un conocimiento integral de arquitectura de software y de los frameworks disponibles para llevar a cabo un adecuado proceso de análisis, diseño y desarrollo de aplicaciones, acorde a las especificaciones y requerimientos empresariales.

Esta investigación es una guía que permitirá al lector y específicamente a personas relacionadas con ingeniería de software, obtener conocimientos sobre:

- Uso de procesos de arquitectura de software en proyectos de desarrollo de software.
- Diseño de arquitectura de software.
- Aplicaciones empresariales.
- frameworks de arquitectura de software y su aplicabilidad.

## ALCANCE

En el presente documento se aborda el tema referente a la arquitectura de software, partiendo de sus antecedentes históricos hasta abordar los diferentes tipos de frameworks, que pueden ser utilizados en el desarrollo de aplicaciones empresariales.

# ARQUITECTURA DEL SOFTWARE

## 1. ANTECEDENTES HISTÓRICOS

Para la implementación de un software se debe tener en cuenta una serie de elementos como lo son los componentes, principios, así como la interacción entre el software y el hardware. Para conocer de forma más explícita a que se refiere la arquitectura de software y tomando como referencias algunos conceptos de wikipedia al respecto, se cita lo siguiente:

> *"La Arquitectura del Software es el diseño de más alto nivel de la estructura de un sistema".*
>
> *"Una arquitectura de software se selecciona y diseña con base en objetivos (requerimientos) y restricciones. Los objetivos son aquellos prefijados para el sistema de información, pero no solamente los de tipo funcional, también otros objetivos como la mantenibilidad, auditabilidad, flexibilidad e interacción con otros sistemas de información. Las restricciones son aquellas limitaciones derivadas de las tecnologías disponibles para implementar sistemas de información".* El origen del estudio se remonta al año 1960.
>
> *En 1968 "EdsgerDijkstra, propuso que se estableciera una estructuración correcta de los sistemas de software antes de lanzarse a programar, escribiendo código de cualquier manera".*

Aunque Dijkstra no emplea en ningún momento el término arquitectura para describir el diseño conceptual del software, sus conceptos estipulan las bases para lo que luego expresarían NiklausWirth (Abril de 1971) como stepwise refinement y De Remer y Kron (Kron 1976) como programming-in-thelarge o programación en grande, idea que poco a poco se irían haciendo objeto de opinión entre los Ingenieros y Arquitectos.

En 1969, en la conferencia de la NATO, P. I. Sharp formuló lo siguiente:

> *"Pienso que tenemos algo, aparte de la ingeniería de software, algo de lo que hemos hablado muy poco pero que deberíamos poner sobre el tapete y concentrar la atención en ello, es la cuestión de la arquitectura de software. La arquitectura es diferente de la ingeniería, ejemplo de lo*

*que quiero decir, echemos una mirada a OS/360. Partes de OS, si vamos al detalle, han utilizado técnicas que hemos acordado constituyen buena práctica de programación. La razón de que OS sea un amontonamiento amorfo de programas es que no tuvo arquitecto. Su diseño fue delegado a series de grupos de ingenieros, cada uno de los cuales inventó su propia arquitectura. Y cuando esos pedazos se clavaron todos juntos no produjeron una tersa y bella pieza de software (Sharp 27 al 31 de Octubre de 1969)."*

En 1972, Parnas publicó un ensayo en el cual discernía sobre la forma en que la modularidad en el diseño de sistemas podía mejorar la flexibilidad y el control conceptual del sistema, acortando los tiempos de desarrollo. De esta manera se introduce el concepto de ocultamiento de información (information hiding), siendo unos de los principios del diseño de software aún en la actualidad.

En 1975, Brooks, el diseñador del sistema operativo OS/360 y Premio Turing 2000, comenzó a utilizar el concepto de arquitectura del sistema para designar "*la especificación completa y detallada de la interfaz de usuario"* aludiendo desde ese momento, que el arquitecto es un agente del usuario (Jr. 1975). Por otro lado, también distinguía entre arquitectura e implementación; mientras aquella decía qué hacer, la implementación se ocupa de cómo.

Es importante señalar que en la década de los setentas fue cuando se comenzó a hablar del diseño estructurado y se dieron a conocer los primeros modelos explícitos de desarrollo de software, basados en una estrategia más lógica, evolutiva y cíclica, esto significó un paso muy importante, ya que se estaba dejando atrás el desarrollo en cascadas, luego surgieron las primeras investigaciones académicas sobre diseños más intensivos, de allí se inicio la separación del diseño y de la implementación, concibiéndose herramientas, técnicas y lenguajes de modelado, que marcarían el camino hacia lo que hoy se conoce como arquitectura de software.

En los ochentas la popularidad de los métodos de desarrollo estructurado, comenzó a decaer, el análisis estructurado considera un sistema desde la perspectiva de los datos que fluyen a través de él, sin embargo en esa época surgió la programación orientada a objetos, una nueva manera de pensar y rompiendo el paradigma de la programación estructurada.

*La OOP supone, no solo un nuevo paso hacia ese fin, sino que además, a nuestro juicio, es el más importante acaecido hasta el momento. Como*

*nos comenta Eckel: "A medida que se van desarrollando los lenguajes, se va desarrollando también la posibilidad de resolver problemas cada vez más complejos. En la evolución de cada lenguaje, llega un momento en el que los programadores comienzan a tener dificultades a la hora de manejar programas que sean de un cierto tamaño y sofisticación." (Bruce Eckel, "Aplique C++", p. 5 Ed. McGrow-Hill).*

A su llegada trajo consigo varias características que se destacan a lo largo de la historia como lo son: Clase, Objeto, Herencia, Encapsulación y Polimorfismo. A finales de esta década se comienza a vislumbrar un nuevo camino y surge el tema de arquitectura de software. Haciéndose notar en diferentes libros de textos, como por ejemplo el trabajo presentado por Perry y Wolf donde ellos proponen la concepción de la arquitectura del software por analogía con la arquitectura de edificios, muchos abusaron de ella y otros la encontraron muy útil. A lo largo de esa década, se dieron a conocer varios trabajos con propuestas relevantes, entre ellas, la programación basada en componentes, el surgimiento de los patrones y estilos, el modelo de 4+1 vistas y lenguajes de descripción de arquitecturas (ADLs), entre otras.

Estos últimos, son lenguajes que permiten la descripción de la arquitectura del software de un sistema, siendo necesarios para exponer abstracciones útiles para modelar sistemas complejos desde un punto de vista arquitectónico, ellos se enfocan en describir primordialmente los componentes de una arquitectura, las interfaces expuestas por los componentes, los conectores utilizados para acoplar los componente mediante las interfaces y sus configuraciones.

Algunos de estos lenguajes proveen herramientas para el análisis, chequeo, parsing (programa que analiza una porción de texto para determinar su estructura lógica), compilación y sintetización de la descripción de la arquitectura objetivo.

Entre las características se mencionan las siguientes, descritas por [Shaw y Garlan, 1996]:

- **Composición:** permiten la representación del sistema como composición de una serie de partes.
- **Configuración:** la descripción de la arquitectura es independiente de los componentes que formen parte del sistema.

- **Abstracción:** describen los roles abstractos que juegan los componentes dentro de la arquitectura.
- **Flexibilidad:** permiten la definición de nuevas formas de interacción entre componentes.
- **Reutilización:** permiten reutilizar tanto los componentes como la propia arquitectura.
- **Heterogeneidad:** permiten combinar descripciones heterogéneas.
- **Análisis:** permiten diversas formas de análisis de la arquitectura y de los sistemas desarrollados a partir de ella.

Los ADLs proporcionan tanto una sintaxis específica como un marco conceptual para modelar la arquitectura de un sistema de software [Vestal, 1993].

Shaw (1995), establece un modelo para especificar la arquitectura del software haciendo uso de abstracciones predefinidas para describir los componentes, las interacciones entre dichos componentes (caracterizadas en este lenguaje como conectores) y los patrones que dirigen la composición de unos con otros para formar sistemas.

El objetivo del lenguaje es servir de vehículo de representación de las abstracciones utilizadas en la práctica por los diseñadores de software, para lo cual los mecanismos de conexión ofrecidos por el lenguaje deben ser variados.

## 2. CONCEPTOS DE LA ARQUITECTURA DE SOFTWARE

Existen una gran gama de definiciones que dan a conocer lo que es una arquitectura de software, aún en estos tiempos, surgen diversas vertientes en lo que se refiere a este tema, entre los cuales se menciona el significado que para David Garlan y Mary Shaw, en su libro "An introduction to Software Architecture", señalan que la Arquitectura es un nivel de diseño que hace foco en aspectos *"más allá de los algoritmos y estructuras de datos de la computación; el diseño y especificación de la estructura global del sistema es un nuevo tipo de problema"*.

La arquitectura del software Se refiere a un grupo de abstracciones y patrones que brindan un esquema de referencia útil para el desarrollo de software dentro de un sistema informático.

En este mismo orden de ideas, podemos encontrar lo que se denomina la definición “oficial” de la arquitectura del software que es la que se puede observar en el documento de IEEE Std 1471-2000, adoptada también por Microsoft, y la define de la siguiente manera:

> *"La Arquitectura de Software es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución."*

En vista de la gran cantidad y variedad de definiciones existentes sobre Arquitectura de Software, Mary Shaw y David Garlan [SG95] proporcionaron una sistematización iluminadora, explicando las diferencias existentes entre las distintas definiciones en función de las clases de modelos. Los autores realizaron las siguientes clasificaciones de los modelos:

- **Modelos estructurales**: este modelo sostiene que la arquitectura del software, está compuesta por componentes, conexiones entre ellos y asi como también la configuración, estilo, restricciones, semántica, análisis, propiedades, racionalizaciones, requerimientos, necesidades de los participantes. Se caracteriza por el desarrollo de lenguajes de descripción arquitectónica (ADLs).

- **Modelos de framework**: Son similares a la vista estructural, pero radica en la estructura coherente del sistema completo, en vez de concentrarse en su composición. Los modelos de framework a menudo se refieren a dominios o clases de problemas específicos, incluye arquitecturas de software específicas de dominios, como CORBA, o modelos basados en CORBA, o repositorios de componentes específicos, como PRISM.

- **Modelos dinámicos**: Destaca la cualidad conductual de los sistemas. Dinámico, es decir, se refiere a los cambios en la configuración del sistema, o a la dinámica involucrada en el progreso de la computación, tales como valores cambiantes de datos.

- **Modelos de proceso**: Se concentran en la construcción de la arquitectura, y en los pasos o procesos involucrados en esa construcción, en este modelo la arquitectura es el resultado de seguir un argumento (script) de

proceso. Esta vista se ejemplifica con el actual trabajo sobre programación de procesos para derivar arquitecturas.

- **Modelos funcionales**: Una minoría considera la arquitectura como un conjunto de componentes funcionales, organizados en capas que proporcionan servicios hacia arriba. Es tal vez útil pensar en esta visión como un framework particular.

Luego de conocer la semántica de lo que se refiere la arquitectura de software, es importante conocer a profundidad algunos de los aspectos que la constituye:

- **Unidad arquitectónica**

  Es conocida generalmente como componente, se trata de una unidad auto contenida ya que se trata de una unidad para todos los propósitos y actividades de la Ingeniería de Software, una unidad arquitectónica aparece siempre rigurosamente encapsulada tanto para el diseño como para el análisis y debe ser suficientemente grande para que sean escalable y manejable los modelos de arquitectura.

- **Vista y perspectiva**

  Basándose en la definición aportada por la IEEE, *"es una especificación de los convenios necesarios para construir y usar una vista",*siendo una vista *"una representación del sistema basada en un conjunto de artefactos relacionados".*

  La definición de vista y perspectiva de un sistema tiene las siguientes ventajas:

  - Promueven la separación de aspectos, mejorando así la eficiencia del proceso de desarrollo.

  - Las inconsistencias pueden surgir de tener vistas diferentes del mismo sistema. Muchas de esas inconsistencias pueden ser detectadas automáticamente, permitiendo la posibilidad de implementar una herramienta con soporte a la depuración del sistema.

- El uso de vistas y perspectivas para una catalogación y recuperación automática de software desde base de datos de componentes.

- **Estilos arquitectónicos**

  Shaw & Clement, definen el estilo arquitectónico como*"un conjunto de reglas de diseño que identifican los tipos de componentes y conectores que pueden ser usados para componer un sistema junto con las restricciones en el modo en que la composición es hecha".*

  Con base a las dos anteriores definiciones se entiende que se establece un vínculo entre el estilo arquitectónico y la unidad arquitectónica.

- **Abstracción**

  La definición que nos proporciona Grady Booch, identifica la abstracción arquitectónica con el encapsulamiento propio de de la tecnología de objetos, "Una abstracción denota las características esenciales de un objeto que lo distinguen de otras clases de objetos y provee de este modo delimitaciones conceptuales bien definidas, relativas a la perspectiva del observador", tanto para la IEEE, Rumbaugh, Shaw, entre otros, la abstracción consiste en extraer las propiedades esenciales o identificar los aspectos importantes o examinar selectivamente ciertos aspectos de un problema posponiendo o ignorando los detalles menos sustanciales o irrelevantes.

- **Proceso arquitectónico**

  Según Sommerville, el proceso de desarrollo de software "es un conjunto de actividades y resultados asociados que producen un producto software". Actualmente existen varios estándares para la mejora del proceso de software (el modelo de capacidad de madurez (CMM)) y los métodos industriales como catálisis (Cook &Daniels [25]) y proceso estándar unificado (UP).

## 3. TIPOS DE ARQUITECTURA DE SOFTWARE DE IT

De acuerdo a la distribución del software en cada nodo de la red dentro de la empresa, se pueden encontrar tres proposiciones de arquitecturas de capas para los Sistemas de Información, a menudo estas capas se conocen como niveles “tiers” las cuales se nombran a continuación:

### 3.1 Arquitectura en capas

Este enfoque tiene su basamento en la repartición de roles y responsabilidades de una manera jerárquica, con el objetivo de suministrar una forma mucho más efectiva para la separación de las responsabilidades. El rol será quien indique el modo y tipo de interacción que tendrá con las otras capas y la responsabilidad indica la funcionalidad que está siendo desarrollada.

- **Características:**
  - Describe la disgregación de los servicios de forma que la mayoría de la interacción ocurra solamente entre capas vecinas.
  - Las capas de una aplicación pueden alojarse en la misma máquina física (misma capa) o puede estar distribuido sobre diferentes computadores (n-capas).
  - Los componentes de cada capa se pueden comunicar con otros componentes en otras capas, para lo cual se utilizan interfaces muy bien definidas.
  - Este tipo de arquitectura se conoce también como “pirámide invertida de re-uso”, debido a que cada capa va agregando responsabilidad y abstracción a la capa que se encuentra sobre ella.
- **Principios fundamentales:**

  Los principios aplicables en el diseño para usar este estilo de arquitectura son los siguientes:

  - **Abstracción:** Separa la vista del modelo como un todo mientras que suministra suficiente detalle que pudiera ayudar a entender las relaciones entre capas.

- **Encapsulamiento:** El diseño no hace exaltaciones acerca de tipos de datos, métodos, propiedades o implementación.

- **Funcionalidad claramente definida:** La separación entre la funcionalidad de cada capa es definido, lo cual conlleva a que capas superiores como la capa de presentación envíen comandos a las capas inferiores como la capa de negocios y la capa de datos, haciendo que los datos puedan fluir hacia y desde las capas en cualquier sentido.

- **Alta cohesión:** Cada capa tiene funcionalidad que se encuentra relacionadas directamente a la tarea de dicha capa.

- **Reutilizable:** Las capas inferiores no tienen ninguna dependencia con las capas superiores, permitiéndoles ser reutilizables en otros escenarios.

- **Desacople:** La comunicación entre las capas está basada en la abstracción lo que provee un desajuste entre las capas.

- **Beneficios**

  Este estilo de arquitectura brinda múltiples beneficios entre los cuales se puede mencionar los siguientes:

  - **Abstracción:** Permite que se realicen cambios en un nivel abstracto, es decir, se puede incrementar o disminuir el nivel de abstracción usado en cada capa de la “pila” jerárquica.

  - **Aislamiento:** Permite recoger los cambios en tecnologías a ciertas capas para reducir el impacto en el sistema total.

  - **Rendimiento:** La distribución de las capas puede incrementar la escalabilidad, la tolerancia a fallos y el rendimiento.

  - **Mejoras en Pruebas:** El tener una interfaz bien definida es de beneficio para llevar a cabo pruebas, así como de la habilidad para cambiar a diferentes implementaciones de las interfaces de cada capa.

- **Ventajas**
    - Se puede lograr reutilizar las capas.
    - Permite que se pueda establecer un estándar de manera muy sencilla.
    - Se puede trabajar en diversos niveles de abstracción,
    - Los cambios pueden limitarse a una capa en particular, es decir, cuando existe un cambio solo es requerido la actualización en la capa que corresponde sin tener que involucrar todo el código y el resto de las demás capas..
- **Desventajas**
    - En ocasiones puede ocurrir que no se logre englobar los cambios a un sola capa, generando una cascada de cambios en el resto de las capas.
    - Puede ser considerada en diversas ocasiones poco eficiente.
    - Posible conflicto en el diseño de los niveles de las capas.
    - Algunas veces puede presentarse trabajo de manera redundante entre una y otra capa.

Por lo anteriormente descrito, se puede concluir que no es necesario inventar una arquitectura de software para el desarrollo de un sistema, existen una gran variedad que puede ser adaptado al desarrollo que se quiere realizar, entre las arquitecturas más comunes, se encuentran:

## 3.2 Cliente-servidor (Arquitectura de dos capas)

Actualmente existen muchos sistemas de información cuya arquitectura se basa en la denominada  dos capas, las cuales solo cuentan con los siguientes niveles o capas:

- Nivel de aplicación.
- Nivel de la base de datos.

Es un modelo de aplicación distribuida en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes. Un cliente realiza peticiones a otro programa, el servidor, es quien le da respuesta. Esta idea también se puede aplicar a programas que se ejecutan sobre una sola computadora, aunque es más ventajosa en un sistema operativo multiusuario distribuido a través de una red de computadoras.

- **Características**

  En esta arquitectura el remitente se conoce como cliente, el cual es quien realiza lo siguiente:

  - Inicia solicitudes o peticiones.
  - Debe esperar y recibir respuestas del servidor.
  - Puede conectarse a varios servidores.
  - Por lo general interactúa con el usuario final a través de una interfaz gráfica.

  En cuanto al receptor de la solicitud enviado por el cliente se conoce como servidor, el mismo se caracteriza por:

  - Desempeñan un papel pasivo en la comunicación, deben esperar a que lleguen las solicitudes.
  - Al recibir una solicitud, la misma es procesada y posteriormente envían la respuesta.
  - Generalmente pueden aceptar las conexiones de varios clientes a la vez.
  - No es frecuente, la interacción directa con el usuario final.

En los anteriores párrafos se ha mostrado los diferentes tipos de arquitectura de software de manera general pero precisa, haciendo énfasis en lo que actualmente se trabaja y ha resultado ser la opción más utilizada como es la arquitectura de software basado en capas, sin embargo, dando una mirada a los tipos de arquitecturas de software de TI (tecnología de la información), allí se agrupan los elementos y las técnicas mas empleadas

en el proceso y transmisión de la información, de allí que se estaría haciendo referencia a los sistemas de información.

Los tipos de arquitectura de software para los sistemas de información son diversos, sin embargo en el siguiente apartado se hará referencia al uso más común en arquitectura de software para TI.

En la actualidad, la arquitectura de software más utilizada, es la arquitectura cuya carga se divide en tres partes (o capas) lo cual permite repartir las funciones: En la capa de presentación, permite la interacción con el usuario, la capa para el cálculo es donde reside la forma del negocio y una última capa que permitirá el almacenamiento de la información.

En toda arquitectura de capa los elementos que se encuentran asociados en una misma capa, pueden realizar una comunicación directa; pero pueden existir diferencias en cuanto a las comunicaciones permitidas entre elementos de capas diferentes:

Este patrón de arquitectura por capas es una de las prácticas que se han vuelto más frecuentes para ser utilizados por los arquitectos de software lo cual permite dividir sistemas de software con el fin de hacerlos menos complicados. En este tipo de diseño la capa que se encuentra más alta utiliza algunos servicios que previamente fueron definidos por la capa inferior, pero la última es ignorante de la superior. Asi mismo, se dice que cada capa encubre las capas que se encuentran en la parte inferior de las siguientes capas superiores a esta.

- **Ventajas**
    - Se puede entender una capa como un todo, sin considerar las otras.
    - Las capas pueden ser sustituidas por implementaciones alternativas de los mismos servicios básicos.
    - Se minimizan dependencias entre capas.
    - Las capas posibilitan la estandarización de servicios.
    - Al tener una capa construida, puede ser utilizada por muchos servicios de mayor nivel.

- **Desventajas**

  - Los ambientes cuya arquitectura se basa en el modelo de dos capas, por lo general demandan un control extremo sobre la generación de las versiones, además requieren de un esfuerzo adicional para la distribución de la aplicación, cuando se le realizan cambios. Todo ello se debe a que generalmente la aplicación lógica reside en cada una de las estaciones de trabajo del cliente.

  - Los sistemas desarrollados bajo este tipo de diseño, es decir, de dos capas suelen ser complejos, es importante considerar la seguridad de los mismos ya que a menudo se requiere acceso a las bases de datos desde dispositivos que directmente pueden llegar al ambiente de esas bases de datos.

  El gráfico que a continuación se presenta representa un esquema tipico de arquitectura definida anteriormente:

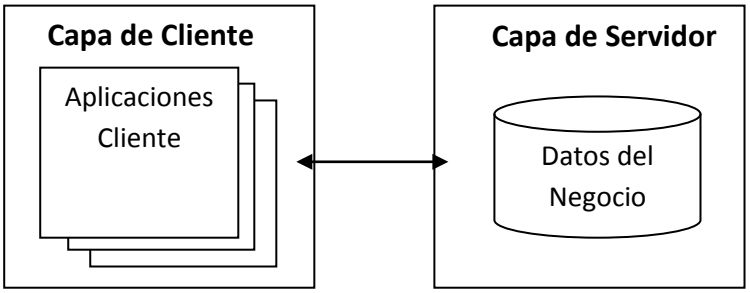



**Gráfico 1. Arquitectura Cliente –Servidor**

## 3.3 Arquitectura en tres (3) capas

- **Características**

  La arquitectura de tres capas se encuentra constituida de la siguiente manera:

- **Capa de Presentación**

  Tiene que ver con la interacción entre el usuario y la aplicación (software). Suele ser tan sencillo como un menú basado en líneas de comando o tan complejo como una aplicación basada en formas. Su objetivo principal es poder ofrecer información al usuario, descifrar los comandos de éste y poder efectuar ciertas validaciones de carácter simples sobre los datos ingresados.

- **Capa de Reglas de Negocio (Empresarial)**

  Conocida como la lógica de dominio, comprende toda la ejecución que se implementa en la aplicación. Abarca los cálculos que se sustentan de la información recibida por el usuario y los datos que se encuentran almacenados y sus respectivas validaciones. Esta capa lleva el control de la ejecución de la capa de acceso a la de datos y servicios externos.

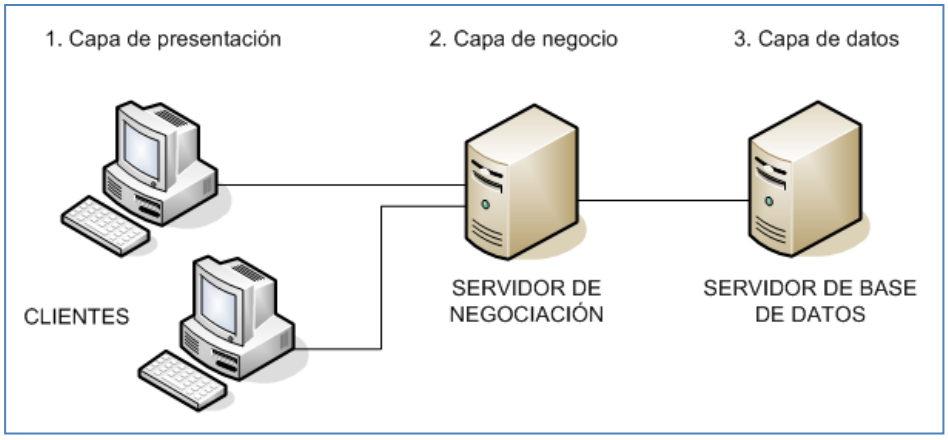
  Por otro lado, se debe realizar el diseño que corresponde a la lógica de la capa de negocios, esto con el fin de conocer el uso por parte de los componentes de presentación o su encapsulamiento como servicio y las llamadas a través de una interfaz de servicios que realiza la coordinación de la conversación con los clientes del servicio o demanda cualquier flujo o componente de negocio.

- **Capa de Datos**

  Engloba la lógica de comunicación con los sistemas que deben realizar operaciones en la aplicación. Las cuales pudieran ser transacciones, otras aplicaciones, sistemas de mensajerías, entre otras.

  Por otro lado, en el caso de las aplicaciones empresariales, está expresado por las base de datos, siendo estas las responsables por el almacenamiento reiterativo de la información. Ademas esta capa debe realizar completamente la separación de las capas superiores (negocio) y del lenguaje utilizado para comunicarse con los repositorios de datos (PL/SQL, Transact-SQL, entre otros).

La siguiente gráfica muestra la represetación de la arquitectura de tres capas.




**[1]Gráfico 2. Arquitectura Tres (3) Capas**

- **Ventajas**

  - No se requiere una interfaz con el usuario final para realizar la comunicación con el receptor de los datos. Por lo consiguiente se puede modificar esa estructura de los datos sin tener que cambiar la interfaz del usuario en el computador.

  - Se puede reutilizar por varias aplicaciones, el código de la capa intermedia siempre y cuando se haya desarrollado o diseñado de forma modular.

  - Es mucho más sencillo realizar el reemplazo o modificación de una capa sin tener que afectar los módulos restantes, esto debido a que los componentes se encuentran de manera centralizado lo cual permite su fácil desarrollo, mantenimiento y uso.

[1] Autora: Milagros Arenas paredes, gráfica referida en la investigación sobre Arquitectura en capas, obtenido de http://arquitecturaencapas.blogspot.com/2011/08/arquitectura-3-capas-programacion-por.html. Arquitectura de n- capas: http://iutll-abdd.blogspot.com/2012/05/arquitectura-de-n-capas.html. Fuentes.

- **Desventajas**

  - Puede haber un incremento en el tráfico en la red y por lo tanto se requerirá más balanceo de carga.

  - El cliente no puede tener a su disposición los recursos que puedan hallarse dispuesto en el servidor.

### 3.4 Arquitectura de N Capas (N- Layer, N-Tier)

- **Características**

En esta arquitectura se comienza a experimentar con una capa adicional:

- Presentación.
- Aplicación.
- Dominio de la aplicación.
- Repositorio.

El objetivo principal es la separación de la programación GUI de la aplicación, en el nivel de la presentación no se realizan cálculos, consultas o actualizaciones sobre el dominio, no se visualiza la capa del dominio. La capa de la aplicación es la encargada de accesar la capa del dominio, simplificar la información del dominio convirtiendo los tipos de datos que comprende la interfaz: enteros, reales, cadenas de caracteres, fecha y clases contenedoras (*container, collection*).

Por otro lado, es importante conocer las diferencias entre capas (layers) y niveles (tiers). [2] Las capas (layers) se ocupan de la división lógica de componentes y funcionalidad en servidores separados, las mismas no tienen en cuenta las topologías de redes así como la localización remota. Sin embargo, los niveles (tiers) usan conjuntos similares de nombres (presentación, servicio, negocios y datos), es importante mencionar que solo los niveles implican una separación física. La siguiente gráfica muestras las diferencias entre las arquitecturas descritas.

[2] Cesar de la torre Llorente, Unai Zorrilla castro, Miguel Ramos, Javier Calvarro, (2010) Obtenido de Guía de arquitectura N-capas orientado al dominio con .net/ capas vs. Niveles. Fuentes.

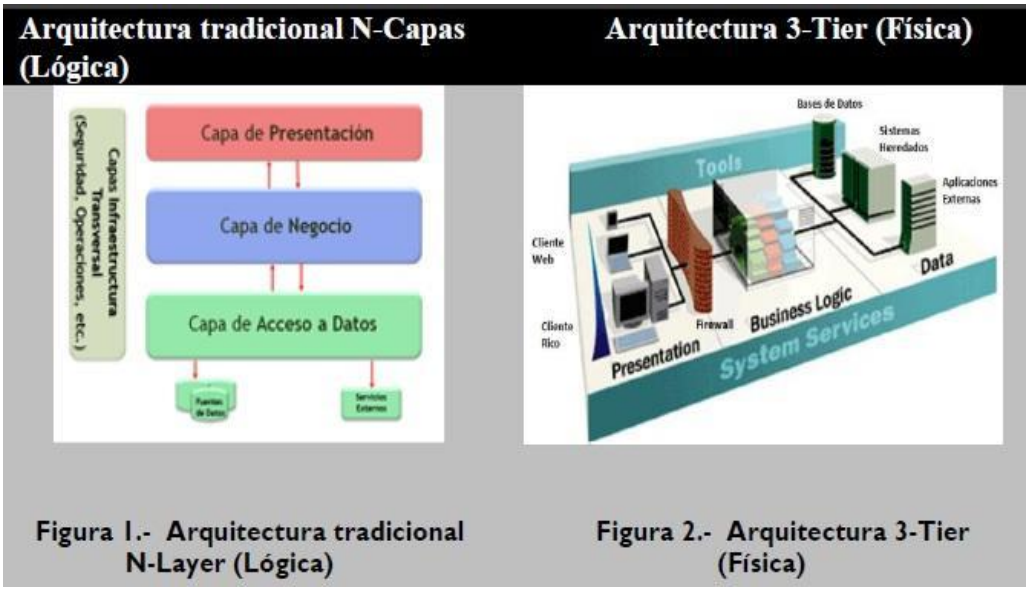



**Gráfico 3. Arquitectura N- Layer, N-Tier**

Las capas son agrupaciones horizontales lógicas de componentes de software que forman la aplicación o servicio.

**Diseño básico de capas**

Se debe realizar la separación de los componentes de la solución en capas, los cuales deben ser cohesivos y aproximadamente el mismo nivel de abstracción.

La clave de una aplicación de n-capas está dada por la gestión de dependencias, en una arquitectura de n-capas tradicional, los componentes de una capa pueden interactuar solo con componentes de la misma capa o con otros componentes de capas inferiores

- **Ventajas**
    - **Flexibilidad:** Proporciona la posibilidad que los componentes puedan ser modificados, estoy con el fin que puedan realizar sus operaciones sin necesidad de recompilar la aplicación, de esta manera se resguarda el contrato definido para la operación. De igual manera permite la reutilización de los componentes en otros tipos de

aplicaciones y no solamente en la aplicación para la que fueron diseñados.

- **Mantenibilidad:** proporciona la tarea de modificar un componente con el fin de corregir errores, optimizar el desempeño, adicionar atributos o adaptarlos a un ambiente variable.[3]

- **Reutilización:** permite que los componentes puedean ser utilizados desde otros componentes o desde otros sistemas. Lo cual quiere decir que inclusive, si los componentes de negocio son consumidos a través de servicios, estos pueden ser reutilizados por otros sistemas internos o externos.

- **Escalabilidad:** es la propiedad que permite que en este caso un componente se pueda adaptar al cambio o puedan crearse nuevos componentes sobre los componentes base para poder especializar más las capacidades de éste, específico para un cliente.[4]

- **Desventajas**

  - Se invierte mucho tiempo en la creación de los componentes "*core"* de los sistemas al inicio del desarrollo y las empresas, por lo general, requieren ver resultados de manera rápida, por lo cual es necesario que se muestre como quedaría el sistema al final del desarrollo.

## 4. PROCESOS DE ARQUITECTURA DE SOFTWARE

Un proceso de arquitectura de software tiene como propósito la producción eficaz y eficiente de un producto software que reúna los requisitos del cliente. Dicho proceso, en términos globales se muestra en el gráfico 4, esta fase se considera netamente intelectual, siendo afectado directamente por la creatividad y juicio de las personas involucradas. Por otro lado, se cree que un

[3] Cesar de la torre Llorente, Unai Zorrilla castro, Miguel Ramos, Javier Calvarro, (2010) Obtenido de Guía de arquitectura N-capas orientado al dominio con .net/ capas vs. Niveles. Fuentes

[4] Cesar de la torre Llorente, Unai Zorrilla castro, Miguel Ramos, Javier Calvarro, (2010) Obtenido de Guía de arquitectura N-capas orientado al dominio con .net/ capas vs. Niveles. Fuentes.

producto software en sí es muy complejo, por lo cual es prácticamente inviable conseguir un 100% de confiabilidad de un programa por pequeño que sea.

A continuación se puede observar el desarrollo de un software en la siguiente gráfica

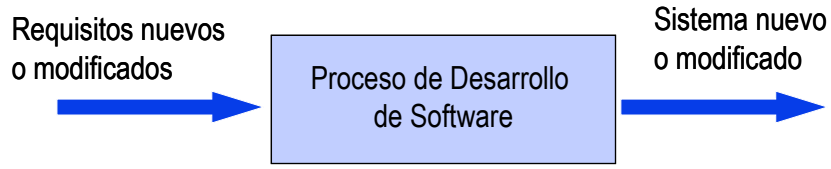



**Gráfico 4. Desarrollo de Software**

El proceso para la arquitectura de software no es único. No existe un proceso de software universal que sea efectivo para todos los contextos de proyectos de desarrollo. Debido a esta diversidad, es difícil automatizar todo un proceso de desarrollo de software.

El comienzo es muy específico, se basa en el planteamiento de la definición de un problema y el alcance del mismo, el planteamiento inicialmente no abarca el tamaño del producto, los riesgos que conllevan su desarrollo y por supuesto los conflictos internos que pudieran presentarse entre los Stakeholders.

Este proceso debe estar sujeto a los siguientes principios:

- Apoyado en las necesidades de los Stakeholders.
- Comunicación de las decisiones.
- El proceso debe estar bien estructurado.
- Pragmático (tiempo, dinero, entre otros).
- Flexible.
- Independiente de tecnología.
- Integrado al proceso de desarrollo.
- Alineado con las políticas de calidad.

Por otro lado, lo principal es obtener una serie de productos, a continuación se hace referencia a ello:

**Principal**

- Documento donde se especifica la arquitectura de software a ser utilizada.

**Secundarios**

- Requerimientos totalmente transparentes.
- Lo que esperan los Stakeholders.
- Opciones de uso de diferentes arquitecturas.
- Criterios de aceptación de la arquitectura.
- Los artefactos para iniciar el diseño de forma detallada.

En el siguiente diagrama se puede observar el proceso de arquitectura de software:

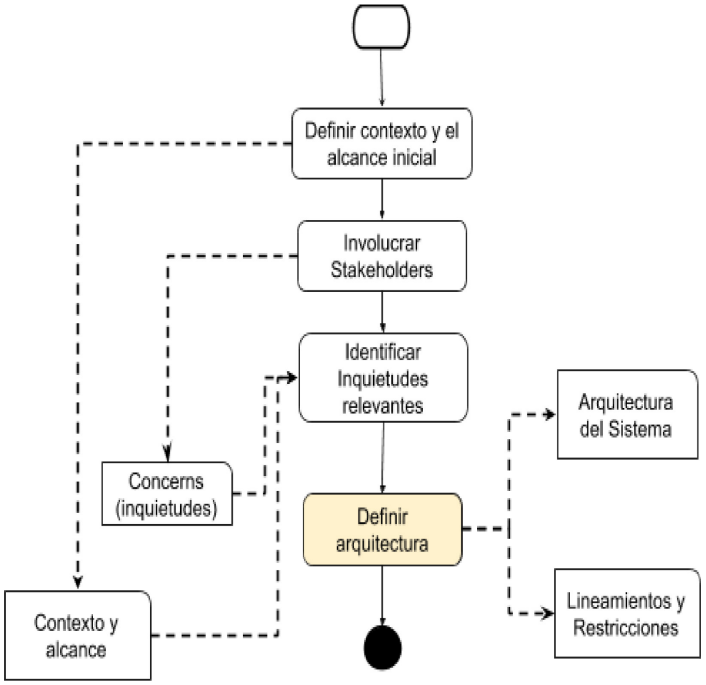



**Gráfico 5. Proceso Arquitectura de Software**

## 5. ALCANCE DE LA ARQUITECTURA DE SOFTWARE

Se debe definir un marco común (estático y dinámico), donde se puede describir todos los elementos importantes para la aplicación, con el objetivo de que sirva como base para la construcción del resto de los elementos del desarrollo. En este sentido y para la comprensión hacia donde debe ir el diseño de la arquitectura y por ende estimar el alcance de la misma, es decir, hasta dónde debe llegar, es por ello que se debe armar un esqueleto que ayude a que todos los elementos del software se mantengan estables durante los numerosos cambios que se producen en su desarrollo.

Definir los límites y el alcance dentro de la empresa es imprescindible para lograr una arquitectura de software adecuada y que se pueda adaptar. Por otro lado, una arquitectura empresarial pragmática, proporciona un contexto y un ámbito. Donde el contexto abarca: las organizaciones, los sistemas y la tecnología que no se encuentra al alcance y que tiene una estrecha relación con la organización. Por otro lado se dice que el arquitecto es sobre quien reside la responsabilidad del enlace entre la aplicación en el contexto y los ingenieros, siendo estos los responsables de los detalles en cuanto al alcance. De igual manera el arquitecto es el responsable de la tarea de los ingenieros y los contratistas para el desarrollo de la aplicación.

## 6. REQUERIMIENTOS Y DRIVERS DE ARQUITECTURA.

Los [5]drivers de arquitectura son un subconjunto de requerimientos que deben ser considerados al momento de establecer la estructuración del sistema, estos requerimientos se pueden categorizar de la siguiente manera:

**Funcionales**: estos engloban todo el requerimiento que se expresan en la conducta de la aplicación y que debe incluir:

- Requerimiento de negocio: Motivo que debe tener el negocio para que exista un sistema.
- Requerimientos de usuario: Comportamiento del sistema, por lo general se expresan en forma de casos de uso.

[5] Humberto Cervantes (Mayo- julio 2010), Requerimientos y arquitectura, obtenido de: http://sg.com.mx/revista/28/requerimientos-y-arquitectura#.VSnhztyG_Sk. Fuentes.

- Requerimientos funcionales detallados: Complementan a los casos de uso (generalmente se describen usando el verbo “deber”).

- Requerimientos de sistema: Describen el mínimo hardware y software para que un sistema de información pueda funcionar.

**No funcionales**: tienen que ver con la manera en que el sistema tolera a los funcionales.

- Reglas de negocio: son todas las normas de la empresa que deben ser llevadas por el sistema.

- Restricciones: son los aspectos que deben considerarse al realizar el diseño y limitan las decisiones que se pueden tomar.

- Interfaces externas: Especificaciones de interfaces de otros sistemas con los que se interactúa.

Por otro lado, se considera que los drivers de la arquitectura incluyen principalmente a los atributos de calidad. Asi como tambien incluyen a un subconjunto de los casos de uso que se consideran como primarios. Los casos de uso primarios son aquellos de mayor importancia o de mayor complejidad para el negocio. Por último, las restricciones también son consideradas como drivers arquitecturales.

## 7. Diseño de arquitectura de software

Abarca el establecimiento de un marco de trabajo estructural, se puede considerar que está compuesta por la estructura jerárquica de los componentes (módulos) y su interacción.

Es importante distinguir las siguientes propiedades a la hora de diseñar una arquitectura de software:

- **Estructurales:** especifica los componentes de un sistema y la forma en que se agrupan en paquetes y su interacción.

- **Extra-funcionales:** se debe indicar como el diseño arquitectónico obtiene los requerimientos de tipo no funcionales como: rendimiento, capacidad, fiabilidad, seguridad, adaptabilidad, entre otras.

- **Familias de sistemas relacionados:** debe permitir reconocer su estructura en los patrones repetitivos que se encuentran de manera habitual en el diseño de sistemas similares. Asi mismo debe ser capaz de reutilizar bloques de construcción arquitectónicos.

Un buen diseño arquitectónico debe ser capaz de describirse utilizando diferentes tipos de modelos, como los que se mencionan en el siguiente apartado:

- **Estructurales:** Representación de la arquitectura semejante a una colección organizada de componentes.

- **Framework:** Se identifican patrones de diseños arquitectónicos repetibles que se encuentran en aplicaciones similares.

- **Dinámicos:** Muestra los aspectos relacionados al comportamiento dinámico de la arquitectura, mostrando los posibles cambios de la estructura y de su comportamientos, todo ello de acuerdo a los eventos externos.

- **De procesos:** Su enfoque es dirigido al modelo del negocio que el sistema deba soportar.

- **Funcionales:** Muestra la jerarquía funcional del sistema.

## 8. DEFINICIÓN DE ATRIBUTOS DE CALIDAD

El Instituto de Ingeniería del Software, plantea que los atributos de calidad sean descritos usando para ello “escenarios”. [6] Un escenario expresa una respuesta medible del sistema ante un estímulo en un contexto particular. Una ventaja de esta técnica de especificación es que un escenario es, automáticamente, un caso de prueba para el sistema.

Por otro lado, se puede observar que otros autores consideran que los atributos de calidad son las características que permiten establecer un concepto que con frecuencia suele ser muy subjetivo, estas deben ser medibles y bajo ciertos

[6] Humberto Cervantes(Mayo- julio 2010), Requerimientos y arquitectura, obtenido de: http://sg.com.mx/revista/28/requerimientos-y-arquitectura#.VSnhztyG_Sk. Fuentes

parámetros acordados previamente con el cliente. Estos parámetros pueden ser: seguridad, confiabilidad, modificabilidad, usabilidad y desempeño, todo ello partiendo de las características del negocio y los objetivos del mismo.

Bass et al. (1998), establece una clasificación de los atributos de calidad en dos categorías:

- Observables vía ejecución: son aquellos atributos que se determinan del comportamiento del sistema en tiempo de ejecución. La descripción de algunos de estos atributos se presenta en la tabla 1.

- No observables vía ejecución: aquellos atributos que se establecen durante el desarrollo del sistema. La descripción de algunos de estos atributos se presenta en la tabla 2.

| Atributo de Calidad | Descripción |
|---|---|
| Disponibilidad | Es la medida de disponibilidad del sistema para el uso (Barbacci et al, 1995). |
| Confidencialidad | Es la ausencia de acceso no autorizado a la información (Barbacci et al, 1995). |
| Funcionalidad | Habilidad del sistema para realizar el trabajo para el cual fue concebido (Kazman et al., 2001). |
| Desempeño | Es el grado en el cual un sistema o componente cumple con sus funciones designadas, dentro de ciertas restricciones dadas, como velocidad, exactitud o uso de memoria. (IEEE 610.12). |
| Confiabilidad | Es la medida de la habilidad de un sistema a mantenerse operativo a lo largo del tiempo (Barbacci et al., 1995). |
| Seguridad externa | Ausencia de consecuencias catastróficas en el ambiente. Es la medida de ausencia de errores que generan pérdidas de información (Barbacci et al, 1995). |
| Seguridad interna | Es la medida de la habilidad del sistema para resistir a intentos de uso no autorizados y negación del servicio, mientras se sirve a usuarios legítimos (Kazman et al., 2001). |

[7]Tabla 1. Descripción de atributos de calidad observables vía ejecución

[7] Joan Arlet Carrascoso P., Enrique Chaviano, Anisleidy Céspedes (7 de Mayo 2009), Procedimientos para la evaluación de arquitecturas de software basadas en componentes, obtenido de: http://www.gestiopolis.com/administracion-estrategia/procedimiento-para-la-evolucion-de-las-arquitecturas-de-software.htm. Fuentes

| Atributo de Calidad | Descripción |
|---|---|
| Configurabilidad | Posibilidad que se otorga a un usuario experto a realizar ciertos cambios al sistema (Bosch et al., 1999). |
| Integrabilidad | Es la medida en que trabajan correctamente componentes del sistema que fueron desarrollados separadamente al ser integrados. (Bass et al. 1998) |
| Integridad | Es la ausencia de alteraciones inapropiadas de la información (Barbacci etal., 1995). |
| Interoperabilidad | Es la medida de la habilidad de que un grupo de partes del sistema trabajen con otro sistema. Es un tipo especial de *integrabilidad* (Bass et al. 1998) |
| Modificabilidad | Es la habilidad de realizar cambios futuros al sistema. (Bosch et al. 1999). |
| Mantenibilidad | Es la capacidad de someter a un sistema a reparaciones y evolución (Barbacci et al., 1995). Capacidad de modificar el sistema de manera rápida y a bajo costo (Bosch et al. 1999). |
| Portabilidad | Es la habilidad del sistema para ser ejecutado en diferentes ambientes de computación. Estos ambientes pueden ser hardware, software o una combinación de los dos (Kazman et al., 2001). |
| Reusabilidad | Es la capacidad de diseñar un sistema de forma tal que su estructura o parte de sus componentes puedan ser reutilizados en futuras aplicaciones (Bass et al. 1998). |
| Escalabilidad | Es el grado con el que se pueden ampliar el diseño arquitectónico, de datos o procedimental (Pressman, 2002). |
| Capacidad de Prueba | Es la medida de la facilidad con la que el software, al ser sometido a una serie de pruebas, puede demostrar sus fallas. Es la probabilidad de que, asumiendo que tiene al menos una falla, el software fallará en su próxima ejecución de prueba (Bass et al. 1998). |

Tabla 2. Descripción de atributos de calidad no observables vía ejecución[8]

[8] Joan Arlet Carrascoso P., Enrique Chaviano, Anisleidy Céspedes (7 de Mayo 2009), Procedimientos para la evaluación de arquitecturas de software basadas en componentes, obtenido de: http://www.gestiopolis.com/administracion-estrategia/procedimiento-para-la-evolucion-de-las-arquitecturas-de-software.htm. Fuentes

# 9. VISTAS Y VIEWPOINTS

## 9.1 VISTAS

EL IEEE (2000) propone construir una plataforma para la representación de arquitecturas de software y confecciona los siguientes términos que se consideran básicos, que son arquitectura, vista y punto de vista.

La vista se puede considerar una herramienta conceptual que permite manejar la complejidad de una arquitectura software, se considera cada vista como un filtro de información, las vistas permiten la representación de uno más aspectos de la arquitectura de software (Normalmente 1 vista = 1 aspecto), siendo estos de tipo estructural y de comportamiento.

El estructural se representa mediante un grafo de nodos, mientras que el de comportamiento a través de un diagrama de estados.

- **Tipo de vistas**
  - **Lógica:** Captura las entidades lógicas (software) de un sistema y cómo se interconectan.
  - **Física:** Describe los recursos físicas (hardware) de un sistema y sus interrelaciones.
  - **De componente:** Los elementos representan las unidades de computación (componentes) y mecanismos de comunicación (conectores) que definen la estructura de desarrollo y ejecución de las aplicaciones.
  - **Comportamiento:** Captura el comportamiento esperado del sistema o partes de éste.
  - **Casos de uso:** Captura los requisitos funcionales que serán ofrecidos por el sistema de información.

- **Ventajas**
  - Separación de problemas.
  - Comunicación con los Stakeholders (actores).
  - Reducción de complejidad.
  - Facilidad en el diseño y desarrollo del sistema.
- **Desventajas**
  - Puede generarse inconsistencia entre las vistas.
  - Ocurrencia de error en la selección de las vistas.
  - Fragmentación de la información.

## 9.2 VIEWPOINTS

Un enfoque o punto de vista (viewpoint) especifica una guía o plantilla (template) para simbolizar un conjunto de responsabilidades (concerns) de acuerdo a una arquitectura, permitiendo la formalización de grupos de modelos, que proporcionan la descripción específica o contenido de una arquitectura. Para identificar dichos puntos de vista se debe considerar lo siguiente:

- **Tipos**
  - **Funcional:** Representa los componentes utilitarios del sistema, sus responsabilidades, las interfaces y las diferentes interacciones que pudieran darse.
  - **Información:** Describe la manera como la arquitectura guarda, manipula, maneja y distribuye información.
  - **Concurrencia:** Describe la estructura de concurrencia del sistema y relaciona elementos funcionales con unidades de concurrencia y cómo serán coordinadas y controladas.
  - **Desarrollo:** Se detalla la arquitectura que resiste el proceso para el desarrollo del sistema.

- **Despliegue:** Representa el ambiente internamente donde el sistema se instalará, incluyendo las dependencias con el ambiente de ejecución, incluyendo una correspondencia de los elementos de software con el ambiente de su ejecución.

- **Operacional:** Describe como el sistema puede ser manejado, dirigido y llevado cuando se encuentre en ejecución en el ambiente de producción.

## 10. EVALUACIÓN DE ARQUITECTURAS DE SOFTWARE

Mucho se ha dicho sobre lo que para unos se conoce como tipo de arquitectura y para otros estilos, sin embargo, la arquitectura del software debe verse como una disciplina que ha alcanzado un grado de madurez.

Existe una diferenciación ya que el estilo arquitectónico define una familia de sistemas relacionados, proporcionando un vocabulario específico del dominio de diseño arquitectónico junto con las limitaciones sobre cómo las partes pueden encajar, su uso le permite a los desarrolladores entender más fácilmente un diseño arquitectural rutinario. Sin embargo se dice que el estilo y la arquitectura nacen en el mismo momento, es decir, son concebidos en el mismo instante.

No se mencionaba en ningún texto lo que se conoce como estilos, Robert Allen y David Garlan [AG92] de la Universidad de Carnegie Mellon se refieren a “paradigmas de arquitectura” y “estructuras de sistemas”, luego se va observando como esos paradigmas terminan siendo lo que se conoce en la actualidad como estilos.

Más tarde y por semejanza con la diseño de edificios se establece que una arquitectura de software se define mediante el siguiente modelo:

Arquitectura= {Elemento, forma, razón}.

- **Beneficios del uso de estilo arquitectónico**

  - Reutilización de diseño, ya el mismo puedes ser utilizado por un conjunto de sistemas relacionados.

  - Reutilización del código.

- Fácil comprensión, debido al uso de estructuras estándar de arquitectura.
- Soporta interoperabilidad de los componentes.

Los estilos se muestran en arquitectura de manera teórica y descriptiva de alto nivel de abstracción, en cambio los patrones en todas partes, suele en ocasiones confundirse estos conceptos, los partidarios de los estilos suelen ser denominados arquitectos, en cambio los que se agrupan en torno a los patrones de denominan ingenieros y diseñadores.

## 10.1ESTILO DE FLUJO DE DATOS

Este estilo se caracteriza por la reutilización y la modificabilidad, siendo esta el estilo apropiado para los sistemas que implementan transformaciones de datos constantemente.

Dentro de estilo se tienen las siguientes arquitecturas: Tuberías y filtros (pipeline) y proceso secuencial en lotes.

### 10.1.1 ARQUITECTURA BASADA EN FILTROS (PIPELINE)

Esta arquitectura radica en la transformación de un flujo de datos en un proceso comprendido por varias fases secuenciales, convirtiéndose la entrada de cada una, la salida de la anterior. Es utilizada en el desarrollo de programas para el intérprete de comandos, ya que se pueden concatenar comandos fácilmente con tuberías (pipe).

Una tubería (pipeline) conecta componentes computacionales (filtros) a través de conectores (pipes), de modo que las computaciones se ejecutan a la manera de un flujo, los datos son transportados a través de las tuberías entre los filtros.

De acuerdo a su claridad y su disposición para absorber una funcionalidad, esta arquitectura se puede considerar idonea para cuando se trata de dar a conocer ideas sobre la formalización del espacio de diseño arquitectónico, igual que el tipo de datos stack lo fue en las especificaciones algebraicas o en los tipos de datos abstractos.

- **Ventajas**
    - Permite comprensión del comportamiento de entrada /salida de un sistema.
    - Facilita el mantenimiento y crecimiento.
    - Permiten realizar análisis de ‘deadlock’ y rendimiento.
    - Facilita la reutilización de transformaciones.
    - Fácil agregar / quitar transformaciones.
    - Relativamente sencillo de implementar, a nivel concurrente o secuencial.
- **Desventajas**
    - Frecuentemente tienden a una organización de procesamiento batch.
    - No son buenos para aplicaciones interactivas.
    - Pueden complicarse al tener que mantener dos flujos separados pero relacionados.
    - Puede ser necesario agregar a los filtros conversión de datos de entrada y salida.
    - Pérdida de performance e incremento de complejidad de los filtros.
    - Es difícil soportar interacciones basadas en eventos.

La siguiente figura muestra este tipo de arquitectura.

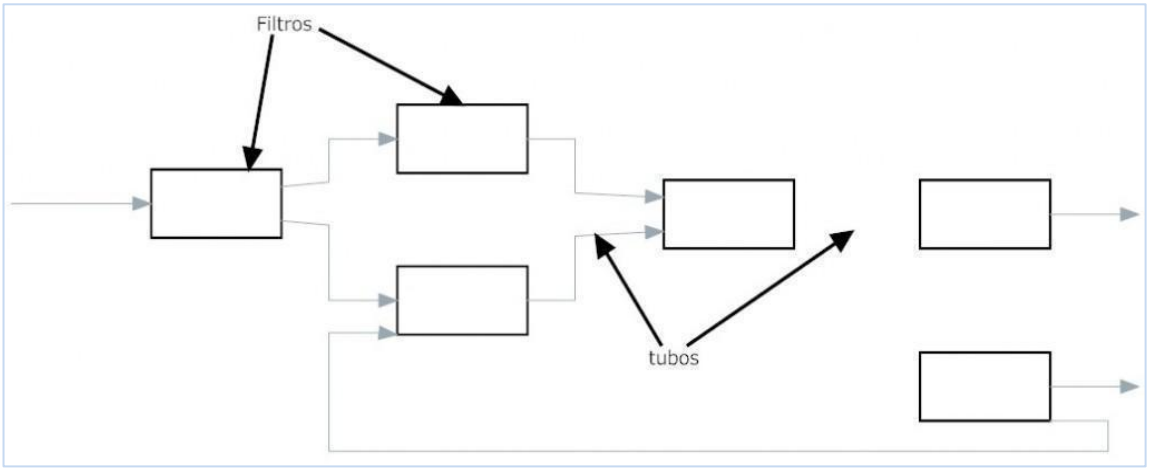



**Gráfico 7. Arquitectura Basada en Filtros**

## 10.2 ESTILOS CENTRADOS EN DATOS

En este estilo se le da importancia a la inserción de los datos, es apropiada para sistemas se origina en el acceso y la actualización de los datos en estructuras de almacenamiento, en este grupo se encuentran: los repositorios, asi como también las bases de datos y las arquitecturas que se basan en hipertextos, en este estilo se incluyen las arquitecturas de pizarra.

### 10.2.1 Arquitecturas de Pizarra o Repositorio

Esta arquitectura está compuesta por: una estructura de datos que constituye la etapa actual y una recopilación de dispositivos independientes que manejan sobre él. Es una muestra arquitectónica de software que son utilizados en sistemas expertos, sistemas multiagente y por lo general, en sistemas que se basan en el conocimiento.

Este tipo de arquitectura, se ha utilizado en aplicaciones que demandan complicadas interpretaciones de proceso de señales como los son el identificación de patrones, comprensión del habla, entre otras, asi como también en sistemas que implican acceso compartido a datos con elementos débilmente acoplado.

El centro de esta arquitectura es un almacén de datos que debe ser accedido frecuentemente por otros componentes para actualizar, adicionar y borrar dichos componentes. Toda arquitectura de este tipo radica en las

subsiguientes secciones: Fuente de conocimiento, que es necesaria para solventar el problema. Una pizarra que simboliza el estado existente para llevar a cabo la resolución del problema. Una táctica, que permite regular la disposición en que deben operar las diferentes fuentes.

Esta arquitectura consta de dos tipos de repositorios que a continuación se mencionan:

- Repositorio pasivo: el cliente en este caso software puede acceder a los datos independiente a cualquier cambio que se produzcan en los datos o en las diferentes funciones de otros clientes considerados software.

- Repositorio activo o blackboard: el repositorio remite la información a los clientes cuando los datos que son de interés muestran cambios en algunos casos, considerándose un ente activo.

- **Ventajas**

  Se dice que esta arquitectura es ventajosa, cuando el problema que se requiere resolver es considerablemente complicado, en cuanto a la parte cognoscitiva. Es decir, cuando el flujo de registro del algoritmo es difícil, o simplemente, se carece de conocimiento del problema que se quiere resolver.

- **Desventajas**

  - No se garantiza la resolución completa del problema.

  - Se considera una arquitectura ineficiente, ya que no se tiene un tiempo de cómputo determinado para llevar a cabo la solución del problema.

## 10.3 ESTILO DE LLAMADA Y RETORNO

En este estilo se hace especial énfasis a la modificabilidad y escalabilidad. siendo los más generalizados en sistemas a gran escala, en este grupo se encuentran las arquitecturas de programa principal y subrutina, los sistemas basados en llamadas a procedimientos remotos, así como también los sistemas

orientados a objetos y los jerárquicos en capas (sobre esta arquitectura se hace referencia, en el apartado de los tipos de arquitectura).

### 10.3.1 Model-View-Controller (MVC)

Son Taylor y Medvidovic, quienes hacen referencia y reconocen este estilo arquitectónico, sin embargo Robert Allen y David Garlan, se refieren a esta, como una microarquitectura, el Model-View-Controller ha sido propio de las aplicaciones Smalltalk, desde 1992 aproximadamente, muchas veces ha sido definido como un patrón de diseño o como práctica recurrente.

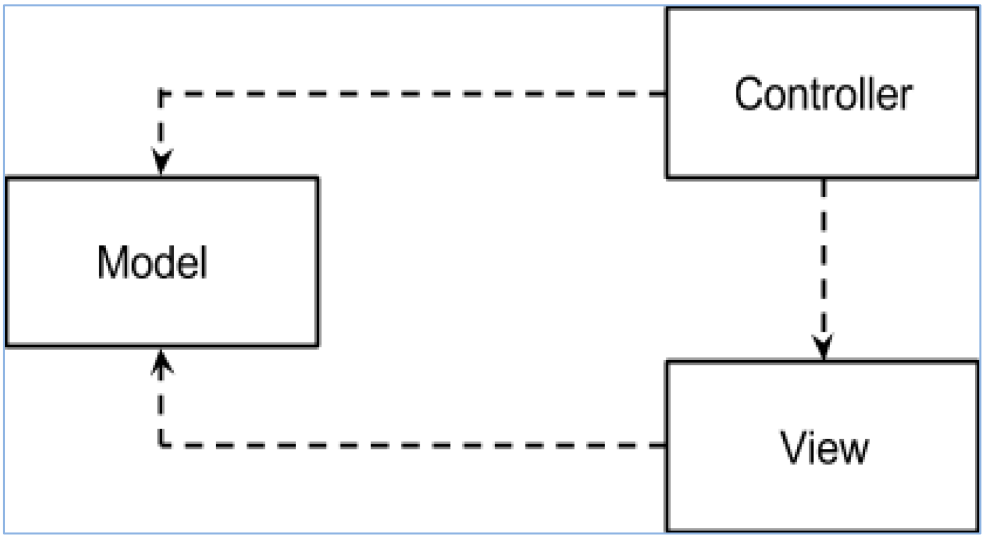



**Gráfico 8. Arquitectura Model-view-controller**

Este modelo realiza una separación entre el modelado y lo que se considera el dominio, asi como la presentación y las diferentes acciones que se basan en los datos ingresados por el usuario en tres clases distintas, esto quiere decir lo siguiente:

- **El modelo:** Administra la actuación y los datos del dominio de aplicación, respondiendo a los requerimientos de información sobre su estado y respondiendo a las instrucciones de cambio de estado cuando así corresponda.
- **Vista:** En ella se maneja la visualización de la información.

- **Controlador:** Es el encargado de interpretar las acciones tanto del ratón, asi como tambien del teclado, asi como de informar al modelo y/o a la vista para que realicen el cambio de acuerdo al requerimiento.

Existen una dependencia hacia el modelo por parte de la vista, como del controlador, sin embargo el no depende de otras clases.

- **Ventajas**
  - Puede soportar vistas múltiples, esto debido a que hay una separación entre el modelo y la vista, esto quiere decir, que la interfaz del usuario permite visualizar múltiples vistas de los datos iguales en forma secundaria.
  - Se adapta a los cambios con facilidad, esto debido a que los requerimientos de interfaz suelen ser cambiantes de manera constante, puede ocurrir que los usuarios prefieran nuevas opciones de representación entre otras y como no existe una dependencia entre la vista y el modelo, realizan estos cambios, se decir, agregar una nueva opción de presentación no afecta el modelo.
- **Desventajas**
  - La complejidad, se profundiza la orientación a eventos relacionados en el código de la interfaz de usuario, llegando a ser difícil de depurar en ocasiones.
  - Generación de costo de actualizaciones frecuentes, esto debido que si el modelo experimenta cambios frecuentes, se podría desbordar las vistas con diversos requerimientos de actualización.

### 10.3.2 Arquitecturas Orientadas a Objetos

Los elementos de este tipo de estilo esta representado por los objetos, considerados también instancias de los tipos de datos denominados abstractos. Para David Garlan y Mary Shaw, los objetos son una representación de un tipo de componentes que ellos denomina managers, los cuales son los responsables de salvaguardar la integridad de su representación. Un aspecto importante es que la representación

de forma interna de un objeto en particular no es alcanzable desde otros objetos.

Se basan en los principios de encapsulamiento, herencia y polimorfismo, también se consideran que las unidades de modelado, asi como el diseño, la implementación, los objetos y las interacciones de estos, son el centro de las atribuciones en el diseño de una arquitectura y de igual manera en la estructura de la aplicación. Las interfaces se hallan separadas de las implementaciones.

Por otro lado, se puede decir que el mejor ejemplo de OO para sistemas distribuidos es Common Object Request Broker Architecture (CORBA), en la cual las interfaces son definidas a través de la Interface Description Language (IDL); un Object Request Broker interviene en las interacciones entre objetos que se consideran clientes y objetos que son los servidores, esto en ambientes conocidos como distribuidos.

Existe una restricción en cuanto a que una interfaz logre ser efectuada por variadas clases. Este estilo puede considerarse a una familia arquitectónica más amplia como la Arquitecturas de Llamada-y-Retorno (Call-and-Return).

- **Ventajas**

  - Se puede cambiar la ejecución de determinado objeto sin causar una afectación a sus clientes.

  - Transformar determinados problemas en una recopilación de agentes que interactuan..

  - Se considera un objeto como una entidad que puede ser reutilizable en el ambiente de desarrollo.

- **Desventajas**

  Se debe conocer la identidad del objeto para poder interactuar con otro utilizando para ello una petición de procedimiento, entre otras limitaciones que posee esta arquitectura.

### 10.3.3 Arquitecturas Basadas en Componentes

Esta arquitectura se fundamenta en los principios que fueron definidos por la ingeniería de software, por otro lado, mucho se ha discutido sobre el significado de componentes, en este sentido Clemens Alden Szyperski han ofrecido una definición que es bastante operativa, y es la siguiente: para ellos un componente de software, es una unidad de composición que contiene interfaces detalladas.

Que sea una unidad de composición y no de construcción quiere decir que no es necesario crear, es decir, se puede obtener hecha, o se puede realizar desde casa con el fin de que otras aplicaciones de la empresa la utilicen en sus propias composiciones.

De igual manera se puede definir un componente de forma pragmática como un artefacto diseñado y desarrollado de acuerdo ya sea con CORBA Component Model (CCM), JavaBeans y Enterprise JavaBeans en J2EE y lo que alternativamente se llamó OLE, COM, ActiveX y COM+, y luego .NET.

- **Ventajas**

  - Las interfaces se encuentran separadas de las implementaciones, siendo estas junto a sus interacciones el centro de incumbencias en el diseño arquitectónico.

  - Los componentes pueden soportan algún régimen de introspección, de modo que su funcionalidad y propiedades puedan ser descubiertas y utilizadas en tiempo de ejecución.

- **Desventajas**

  - Una interfaz sea implementada por múltiples componentes.

  - Los estados de un componente no son accesibles desde el exterior

Es importante hacer mención, que el marco arquitectónico estándar para la tecnología de componentes está constituido por los cinco puntos de vista de RM-ODP (Empresa, Información, Computación, Ingeniería y Tecnología).

## 10.4 ESTILOS DE CÓDIGO MÓVIL

### 10.4.1 Arquitectura de Máquinas Virtuales

La arquitectura de máquinas virtuales se ha llamado también intérpretes basados en tablas, está formado por cuatro componentes como lo son:

- Un motor de simulación o interpretación.
- Una memoria que contiene el código a interpretar.
- Una representación del estado de la interpretación.
- Una representación del estado del programa que se está simulando.

- **Ventajas**
  - El uso de esta arquitectura por lo general es para solucionar un problema de hardware, es decir, que no se disponga del mismo pero se requiere simular uno.
- **Desventajas**
  - En todos los casos, no aplica esta solución de software.
  - Se reduce a lenguaje de programación

## 10.5 ESTILO PEER-TO-PEER

Conocida también como arquitectura de componentes independientes hace énfasis de manera categórica a la modificabilidad por intermedio de los variados segmentos que pudieran intervenir.

Consiste en procesos autónomos u objetos que se pueden comunicar a través de mensajes. Cada entidad puede enviar mensajes a otras entidades, pero no puede controlarlas directamente. Los mensajes pueden ser enviados a componentes nominados o propalados mediante broadcast.

Es importante hacer mención sobre la elaboración de una taxonomía de los estilos basados en transferencia de mensajes, cuya autoría es de Gregory

Andrews, en su estudio y posterior elaboración, señala e identifica ocho categorías, que se mencionan a continuación:

- Flujo de datos en un sentido utilizando para ello redes de filtros.
- Requerimientos y respuestas entre clientes y servidores.
- Interacción de ida y vuelta o pulsación entre procesos próximos.
- Pruebas y ecos en grafos incompletos.
- Broadcasts entre procesos en grafos completos.
- Tokenpassing asincrónico.
- Coordinación entre procesos de servidor descentralizados.
- Operadores replicados que comparten una entidad o tarea.

Dentro de este estilo se encuentra también:

### 10.5.1 Arquitecturas Basadas en Eventos

Estas arquitecturas son conocidas también de invocación implícita e integración reactiva o difusión (broadcast) selectiva, existen diversas estrategias que se pueden conseguir sobre la programación de eventos de acuerdo a las necesidades o requerimientos que se deban cubrir o satisfacer. Este tipo de arquitectura tiene una relación efectivamente con sistemas que se basan en actores, asi como daemons y las redes de intercambio de paquetes, como lo son publicaciones y suscripciones. Los conectores se incluyen procedimientos de llamada tradicionales y vínculos entre informes de incidentes y peticiones a procedimientos.

Los componentes de este estilo de invocación implícita son módulos cuyas interfaces pueden proveer una colección de procedimientos similar al del estilo de tipos de datos abstractos, como un conjunto de eventos, los mismos pueden invocarse de manera usual en modelos orientados a objeto.

Este estilo suele ser utilizado en ambientes de integración de herramientas, así como en sistemas de servicio de base de datos para asegurar las restricciones de consistencia (disparadores), en las

interfaces con el usuario cuyo objetivo principal es la separación de la exposición de los datos de los formas que gestionan datos, de representación similar en editores sintácticamente orientados con el fin de verificar la semántica incremental, entre otros.

- **Ventajas**

  - Contribuye al mantenimiento, esto debido a que los procesos referidos al negocio y que no se encuentran relacionados sean independientes.

  - Se estimula el desarrollo en paralelo, lo cual pudiera de manera positiva contribuir en mejoras de performance.

  - Se considera sencillo de empaquetar en una transacción atómica.

  - Es incrédula en lo que se refiere a si las implementaciones se ejecutan de forma sincrónica o asincrónicamente esto debido a que no se espera una respuesta.

  - Permite agregar y/o reemplazar un componente registrándose para los eventos del sistema.

- **Desventajas**

  - No permite la construir de respuestas complejas a funciones de negocios.

  - Una unidad no consigue manejar los datos o el estado de otra unidad para realizar su tarea.

  - Se puede presentar problemas de performance global y de manejo de recursos, cuando se comparte un repositorio común para coordinar la interacción.

### 10.5.2 Arquitecturas Orientadas a Servicios

Conocida también como SOA, estas arquitecturas basadas en servicios podrían acoplarse con lo que Garlan & Shaw han definido como el estilo de procesos distribuidos. De igual manera otros autores hablan de

Arquitecturas de Componentes Independientes que se comunican utilizando para ello mensajes, en este tipo de arquitectura se puede contemplar dos sub-tipos de este estilo:

- **Participantes especificados (named):** el más destacado es el modelo denominado cliente-servidor. Si el servidor realiza su trabajo de forma sincrónica, va a retornar el control al cliente junto con los datos correspondientes; pero si lo realiza de manera asincrónica, sólo retornará los datos al cliente, manteniendo su propia secuencia de control.

- **Participantes no especificados (unnamed):** Paradigma publish/ subscribe o estilo de eventos.

Este estilo redefine los viejos modelos de ORPC 39 siendo propios de las arquitecturas orientadas a objetos y componentes, estableciendo un modelo en el que se pudiera pensar que cualquier objeto de computo (procedentemente o participando en un wrapper) lograría alcanzar la integración con otra entidad similar a la descrita anteriormente, esto luego de ser resueltas las pertinentes coordinaciones de ontología.

Cuando se habla de un servicio se puede ver como una entidad de software que encapsula la funcionalidad de negocios y facilita esta funcionalidad a otras entidades utilizando para ello interfaces públicas bien definidas. El mismo puede recibir requerimientos originado de cualquier parte, demás su funcionalidad puede ser ampliada y modificada en cualquier momento que así se requiera. Como toda arquitectura tiene sus ventajas y desventajas, sabiendo que es una arquitectura emergente, sin embargo a continuación se mencionan algunas de las ventajas que proporciona esta arquitectura.

- **Ventajas**

  - Mejora la toma de decisiones, esto debido a que logra integrar el acceso a los servicios e información del negocio a un conjunto de aplicaciones dispuestas y combinadas.

  - Mejora la producción en los empleados, ya que se logra un acceso óptimo a los sistemas e información a través de web, cliente avanzado, dispositivo móvil.

- Fortalece la relación con los clientes y los proveedores, se puede obtener mejoras en la capacidad de respuesta para los clientes, para lo cual se puede habilitar portales unificados de los diferentes servicios.

- Ofrecerle a los clientes y proveedores externos acceso a aplicaciones y servicios de negocio dinámicos, esto permite incrementar la satisfacción de los mismos.

Por otro lado, se considera que SOA permite también la documentación del modelo de negocio de la empresa y a utilizar este modelo para integrarse a él de esta manera se estaría dando respuesta a las dinámicas de cambio que se puedan producir y asimismo a optimizarlo.

- **Desventajas**

  - Esta arquitectura depende de la implementación de los patrones o estándares, sin ellos, se considera que la comunicación entre las diferentes aplicaciones requieren de considerable tiempo asi como tambien del desarrollo de código.

  - No se considera para aplicaciones que contienen un alto nivel de transferencia de datos, asi como para aquellas aplicaciones que no demandan la implementación del tipo request/response y mucho menos para aplicaciones que poseen un breve tiempo de vida.

  - Se considera difícil y costoso, esto debido a que debe estar preparado para el cumplimiento de los protocolos y comunicarse con un servicio.

  - Involucra el conocimiento de los procesos que representan el negocio, asi como clasificarlos, obtener las funciones que se consideren comunes entre ellos, colocar estandares y crear las respectivas capas de los servicios que serán solicitadas por los diferentes procesos del negocio.

  - Un servicio aumentara su nivel de criticidad en la medida en que el mismo se incorpore como parte de la descripción de los procesos de negocio. Lo cual implica que cada vez que sea requerido realizar una actualización en ese servicio, como por ejemplo, un cambio en el código, entre otros, se deberá estimar previamente su impacto, sin

embargo, parte de la situación planteada anteriormente, puede ser resuelta con un excelente diseño del servicio.

### 10.5.3 Arquitecturas Basadas en Recursos

Roy Fielding, en el año 2000, expuso su consideración que este estilo denominado también Representational State Transfer o REST, es la consecuencia de la disposición de varios estilos considerados más básicos, que incluyen el repositorio replicado, la cache, el cliente-servidor, los sistemas basados en capas, el sistema sin estado, la máquina virtual, el código a demanda, asi como también la interfaz uniforme, por lo anteriormente mencionado se puede concluir que REST define los recursos reconocibles y técnicas para poder acceder y operar el estado de los mismos.

Por otro lado, Fielding considera que HTTP con su conjunto mínimo de métodos y su semántica sencilla, hace suficientemente general para que se pueda modelar cualquier dominio de aplicación.

En este sentido, se considera claramente que los recursos son unidades de la Web y cada uno de ellos obedece a un protocolo, estos recursos poseen tres aspectos considerables como lo son:

- Conocen cómo se personifica asi mismo al consumidor.
- Saben cómo hacer una conversión de un estado a otro.
- Conocen cómo autodestruirse.

La Arquitectura orientada a recursos define cuatro propiedades que un sistema adherido a ella debe tener:

- **Referencialidad (Addressability):** se debe presentar la información a través de una URI.
- **Carencia de estado (statelessness):** cada solicitud HTTP debe realizarse de manera independiente de las otras, por lo que debe contener toda la información necesaria para ser llevada a cabo.

- **Conectividad (connectedness):** Ésta exige que los recursos no vivan de manera aislada, sino que establezcan enlaces entre ellos, entregando al cliente los estados vecinos al actual y posibilitando así la navegación.[9]

- **Ventajas:**
  - Todos los recursos saben cómo se personifican a sí mismo delante del usuario.
  - El recurso conoce cómo llevar a cabo una conversión desde un estado a otro.
  - Los recursos conocen como autodestruirse.

- **Desventajas:**
  - No es una arquitectura con mucha demanda,.
  - No es considerada para desarrollos web, es decir, no esta definida para ello.

### 10.5.4 REST(Representational State Transfer)

Es un estilo arquitectónico para sistemas hipermedia distribuidos como aquellas aplicaciones para web. REST queda definido por cuatro condiciones de interfaz: identificación de recursos, manipulación de recursos mediante representaciones de estos mensajes auto-descriptivos e hipermedia como motor del estado de la aplicación.

Es importante señalar que REST no es un modelo, se considera solo un estilo de arquitectura. Sin embargo se basa en los siguientes estándares:

- HTTP
- URL
- Representación de los recursos: XML/HTML/GIF/JPEG/, entre otros.

[9] Jonathan forero (Junio, 2013), Consideraciones para una arquitectura interoperable de repositorios de objetos de aprendizaje. Obtenido de: http://posgrado.frba.utn.edu.ar/investigacion/especialidades/forero-2013_tf_esp.pdf

- Tipos MIME: text/xml, text/html, entre otros.

La finalidad de REST es exponer recursos a través de URIs y HTTP, no servicios a través de interfaces de mensajería. Por lo tanto, no debe confundirse con otros protocolos basados en RPC como SOAP O XML-RPC.

- **Ventajas:**

  - Escalabilidad de la interacción con los elementos: La Web ha crecido exponencialmente sin degradar su rendimiento.

  - Generalidad de interfaces: Gracias al protocolo HTTP, cualquier cliente puede interactuar con cualquier servidor HTTP sin ninguna configuración especial.

  - Puesta en funcionamiento independiente: HTTP permite la extensibilidad mediante el uso de las cabeceras, a través de las URLs, a través de la habilidad para crear nuevos métodos y tipos de contenido.

  - Compatibilidad con componentes intermedios: Los más populares son varios tipos de proxys para Web, entre los cuales se tienen, las caches, utilizadas para mejorar el rendimiento. Otros permiten reforzar las políticas de seguridad: firewalls.

- **Desventajas:**

  - En un esquema REST se puede tener varios servidores donde unos no saben que los otros existen.

  - No se sabe si un usuario ha iniciado sesión en un servidor determinado, además no sabe realmente en cual servidor pudo caer una solicitud.

  - No hay un estándar en sus respuestas por lo que no se definen tipos de datos.[10]

[10] Héctor Saira (9 de Agosto 2014) Rest, obtenido de: https://webservicesdsd.wordpress.com/tag/tecnologias/

## 11. APLICACIONES EMPRESARIALES

Las aplicaciones empresariales son software desarrollados para la administración de las operaciones, así como también de los activos y recursos de una empresa.

Las aplicaciones empresariales tienen en general las siguientes características:

- Involucran persistencia de datos.
- Se manejan grandes cantidades de datos.
- Existen varias interfaces de usuario, para distintos tipos de usuario.
- En general se deben integrar con otras aplicaciones.
- Se accede a los datos de forma concurrente.

En el proceso de desarrollo de una aplicación empresarial se involucran los siguientes actores:

- Programadores de aplicaciones.
- Administradores de base de datos.
- Diseñadores de interfaz de usuario.
- Integradores de aplicaciones.
- Arquitectos técnicos y arquitectos de solución.
- Analistas funcionales
- Líderes técnicos.

### 11.1 Tipo de Aplicaciones Empresariales:

- Según el origen, pueden ser: aplicaciones estándar, configurables y a la medida.
  - **Aplicaciones estándar:** Son software desarrollados con la objetivo de que sean utilizados por diferentes empresas y no se permite la

modificación por el usuario. Por otra parte, la empresa fabricante no considera aspectos particulares de sus clientes al desarrollar nuevas versiones y incorporando funcionalidades acorde y adaptable a cualquier empresa.

- **Aplicaciones configurables**: Son software desarrollados de forma tal que permita que el usuario pueda realizar la configuración de muchas de las opciones. Los más conocidos son: ERP (Planificación de Recursos de la Empresa), SAP, People, Soft, Navision y CRM (Manejador de las relaciones con el cliente) como Oracle Siebel o Microsoft Dynamics.

- **Aplicaciones a la medida:** Aplicaciones que son desarrolladas de manera exclusiva para un cliente determinado. El cual puede participar en la fase de análisis y en la posterior creación del programa, es decir, se toma en consideración los requisitos suministrados por el mismo para el desarrollo.

- Según su función: aplicaciones para llevar el control de las compras, lo relacionado a recursos humanos, almacén y la contabilidad, entre los cuales se puede mencionar:

  - **Aplicaciones sectoriales:** Son parte de las aplicaciones estándar, su desarrollo esta orientado específicamente a una empresa de un sector en particular.

  - **Aplicaciones para el control de compra/venta:** Son desarrollos cuyo destino es llevar el registro de los presupuestos, asi como tambien de los pedidos, del material recibido, las diferentes notas de entrega, todas las facturas y ocasionalmente los diferentes cobros y los respectivos pagos. Por lo general son integradas con los sistemas destinados para llevar el control de almacén y la contabilidad.

  - **Aplicaciones de gestión de recursos humanos:** Software para llevar lo correspondiente al personal de la empresa, es decir, para cargar la información del trabajador y posteriormente la generación de la nomina.

  - **Aplicaciones para el control del almacén:** El objetivo del desarrollo de estas aplicaciones es garantizar que se dispone de los productos

precisos en cada momento, para lo cual se debe gestionar las entrada/salida de los diferentes productos, las reservas, los pedidos pendientes por entregar y por recibir, asi como también la información estadística para realizar previsiones.

- **Aplicaciones de contabilidad:** Son programas que se desarrollan para cargar el registro de lo correspondiente a la variación de la realidad financiera de la empresa. En estas aplicaciones los datos que se deben registrar y los informes que deben ser emitidos son acorde con las disposiciones de la ley.

## 11.2 Metodologías para el desarrollo de aplicaciones empresariales

Existen muchas metodologías que ayudan o facilitan el diseño y desarrollo de este tipo de aplicaciones, entre las cuales se puede mencionar la metodología “TOGAF”, esta metodología nace justamente de la necesidad de que tienen las empresas de implementar una solución informática integral, permitiendo la integración de las **cuatro capas fundamentales** para el desarrollo de sistemas de información como son:

- El negocio,
- Los datos,
- Las aplicaciones,
- La tecnología TI.

Esto debido a que las metodologías tradicionales que aún se utilizan no permiten realizar esta integración, debido a que sus componentes no se comunican entre sí, ya que cada componente trabaja de forma independiente.

Con TOGAF se busca definir y especificar un modelo para la arquitectura empresarial de software altamente distribuida con la obligación de definir los estándares, la especificación y la evaluación de requerimientos.

Se basa en los cuatro pilares ya mencionados su integración se realiza en el desarrollo del proyecto, cada una de ellas se cree un sub-sistema de la arquitectura en cuestión, lo que facilita la administraciòn de sus componentes, las interacciones y los objetivos en el tiempo

### 11.2.1 Tipos de las arquitecturas soportados por TOGAF

- **Arquitectura de Negocio:** Especifica las tácticas y los procesos que se consideran son claves para el negocio.

- **Arquitectura de Datos:** Determina cómo se debe administrar los diferentes datos correspondientes del negocio.

- **Arquitectura de Aplicaciones:** Define un diagrama para cada uno de los sistemas de aplicación, donde se detalla las diferentes interacciones que involocra los procesos definidos del negocio.

- **Arquitectura Tecnología (TI):** Representa los componentes del hardware, el software, las comunicaciones y de las redes que son necesarios para tolerar el núcleo del negocio.

## 11.3 Métodos de desarrollo de la arquitectura empresarial

Conocido también como ADM, (Architecture Development Method), es el plan determinado por TOGAF para guiar el desarrollo de la arquitectura empresarial garantizando el cumplimiento con las requerimientos empresariales, asi como lo referido a la TI (tecnología de la información) de la empresa. El cual puede ser ajustado y individualizado según lo requerido por la propia organización y luego de ser definido se puede emplear para la gestion de las procedimientos que se debe realizar para llevar a cabo el desarrollo de la arquitectura empresarial.

Seguidamente se muestra el flujo de procesos de la arquitectura en cuestión:


Gráfico 9. Flujo Arquitectura Empresarial

Como se puede observar, el proceso se vuelve iterativo y cíclico e inicia en la verificación de los requerimientos. Además en la fase C se incluye una composición de la Arquitectura de los Datos y la Arquitectura de las Aplicaciones.

Se considera que cualquier información adicional que se pueda recoger entre los pasos B y C, permitirán el perfeccionamiento de la arquitectura de información.

Algunos de los beneficios que lleve a la utilización de este tipo de arquitectura son:

- Bajos costos de desarrollo.

- ROI: Retorno de la inversión, las inversiones en las nuevas soluciones de sistemas y en la renovación del negocio son recuperadas más rápidamente.

- Reducción de costos: contribuye a la reducción y por ende al retorno de la inversión más rápido.

- Mejorar las relaciones de los departamentos, actores involucrados en pro de unificar criterios para alcanzar los objetivos generales del negocio.

- Respaldo a la inversión: la metodología de TOGAF permite llevar a cabo el desarrollo en la arquitectura y mejorar las inversiones en Tecnología (IT).

- Reducción del Riesgo

## 12. ARQUITECTURA DE SOFTWARE EMPRESARIAL

La IEEE define a la arquitectura como: “The fundamental organization of a system embodied in its components, the irrelationships to each other, and to the environment, and the principles guidingits design and evolution”[38], entonces es necesario ver a la organización como un sistema para poder aplicar esta definición, de esta manera se podría conseguir a un concepto de arquitectura empresarial al definir sus componentes, procesos de negocio, tecnologías y sistemas de información para después establecer sus relaciones y así poder determinar en qué estado se encuentra la organización en el momento que se desee realizar una revisión o un cambio de rumbo.

El objetivo principal es el de unificar los sistemas de hardware y software para todas las unidades de negocio a lo largo de la empresa, relacionados con la parte comercial y que representa normalmente el noventa por ciento (90%) de la organización, al menos en términos de presupuesto.

Todo ello en base de impulsar un cambioestratégico expresados a través de la información generada.

## 12.1 COMPOSICIÓN DE LA ARQUITECTURA EMPRESARIAL

La Arquitectura Empresarial se puede descomponer en cuatro capas o niveles, según lo han determinado los frameworks, los cuales se pueden definir como una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concretos, que puede servir de base para la organización y desarrollo de software[11].

Estas cuatro capas o niveles son consideradas las más usuales y se pueden visualizar en la siguiente gráfica. El orden en que se van a representar, puede no ser el mismo para todas las organizaciones, esto depende si se comienza desde los procesos de negocio hasta el diseño tecnológico o al contrario.

Se considera que la capa más importante es la de Arquitectura de Negocio, debido a su manejo del tema de la sección anterior, procesos de negocio, además por manejar información procedente de las demás capas, es la última capa en ser revisada.

[11] Riehle, Dirk (2000), Framework, obtenido de: http://es.wikipedia.org/wiki/Framework

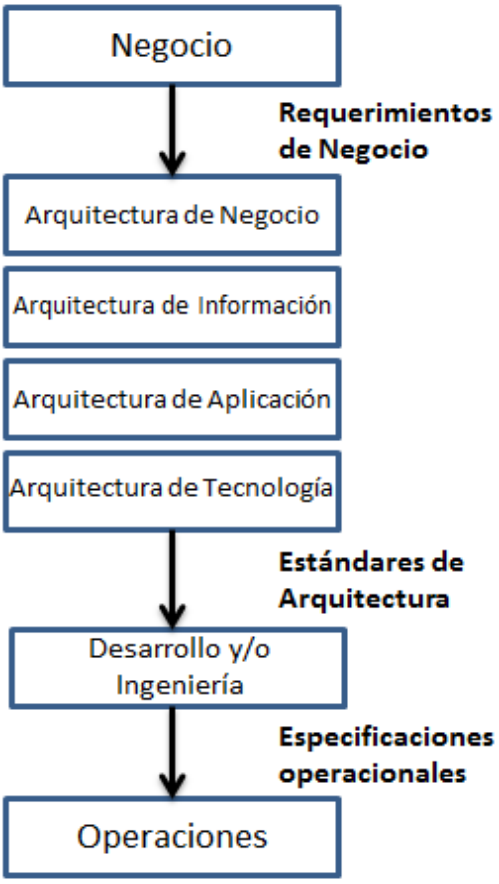



**Gráfico 10. Arquitectura empresarial**

### 12.1.1 Arquitectura de negocio

En esta arquitectura se realiza la definición de la estrategia global de la empresa en base a los cambios que se someterá de acuerdo a los requerimientos que se definen con los stakeholders en un previo proceso de ingeniería de requerimientos.

Se debe considerar oportunamente de manera completa y correcta el estado actual de la organización, identificando las fallas que existen o lo que se debe mejorar, para determinar a qué situación se quiere llegar, es por ello que tanto el enfoque como la estrategia, deben estar muy claros para todas las personas que trabajan en la organización, desde los directivos hasta los desarrolladores, motivo por el cual deben existir motivaciones que lleven a que todos trabajen hacia el logro de un objetivo en común.

### 12.1.2 Arquitectura de Información

Se puede considerar como la integración de una arquitectura de datos, que maneja la información física y lógica, en la cual se puede ver la representación de los mismos en las diferentes vistas, en esta arquitectura se identifican los bloques más importantes de información y los almacena en lugares donde puedan ser consultados de forma más común.

En la siguiente gráfica se representa la arquitectura de información:

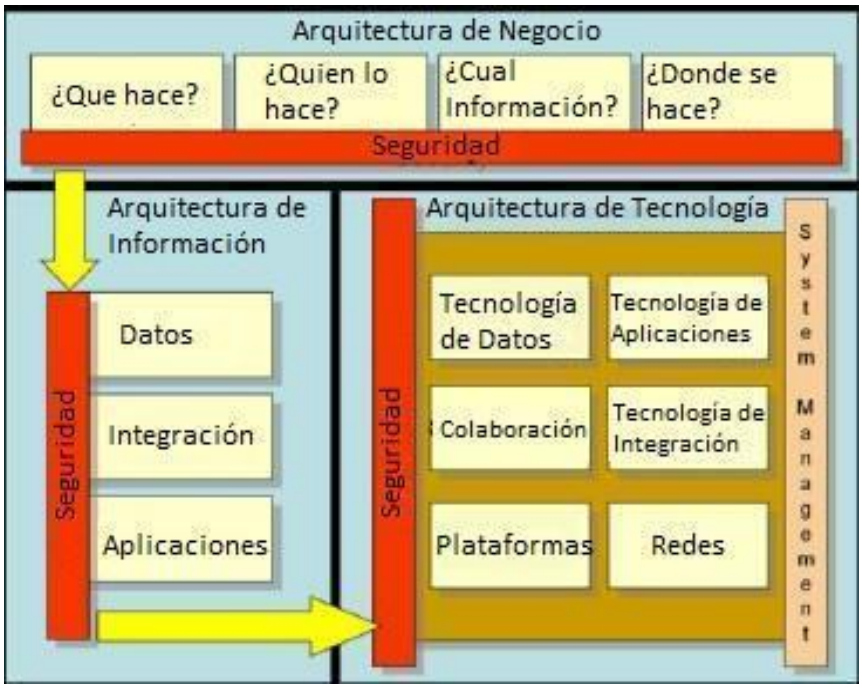



**Gráfico 11. Arquitectura de Negocios**

### 12.1.3 Arquitectura de Aplicaciones

Se denomina también Arquitectura del sistema o de solución, en esta se maneja las funcionalidades y aplicaciones que deben ser desarrolladas de manera individual para luego relacionarlas directamente con los procesos de negocio, considerando las necesidades de la empresa. Se encuentra directamente relacionada con la arquitectura de tecnología ya que se considera un complemento, la continua y genera un mapa de las relaciones entre las aplicaciones de software.

Se encarga de los aspectos técnicos integrales del proceso de creación de productos, desde el requerimiento hasta la implementación, además de la visión técnica integral y la sinergia en el proceso de políticas y planeación, esto se puede ver con más claridad en la siguiente representación gráfica.

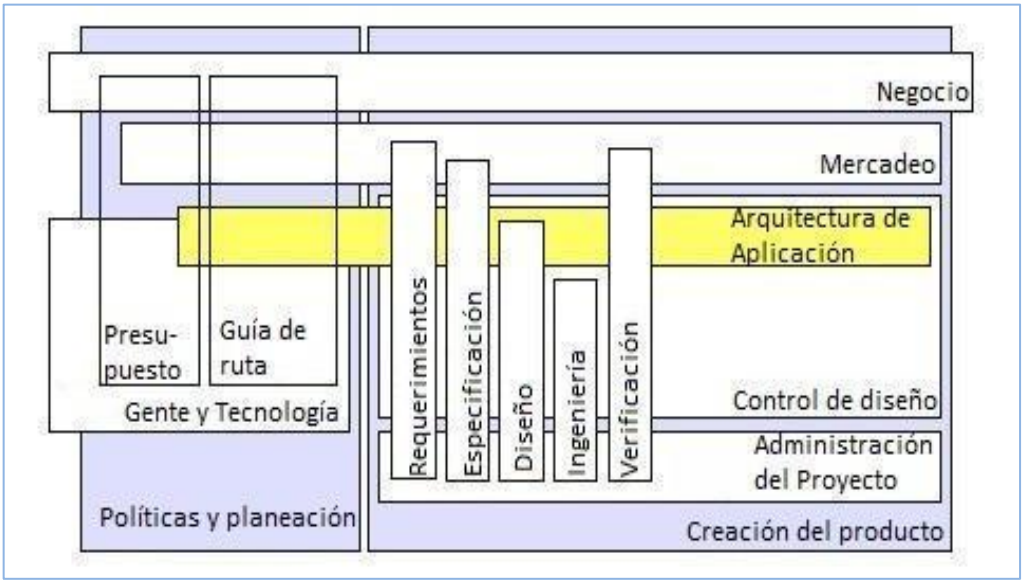



**Gráfico 12. Arquitectura de Aplicaciones**

### 12.1.4 Arquitectura de Tecnología

Es la capa más crítica y por lo tanto más difícil de implementar. En este nivel se reúne los componentes de más bajo nivel dentro de una organización, como el software y hardware que soportan los recursos de bases de datos, directorios, aplicaciones, procesos de soporte, entre otros.

Representa la parte física, la implementación de la solución a la que va a ser sometida la organización. Según The Open Group, basado en su implementación en el ADM de su framework TOGAF, también está fuertemente relacionada con el manejo de las migraciones.

## 12.2 Niveles de madurez de la Arquitectura Empresarial

La Arquitectura Empresarial dentro de una organización dispone de niveles de madurez, en base a su implementación y desarrollo. Estos niveles dependen del avance de la implementación de un framework en la organización, en la medida de su avance en el nivel de madurez, la

previsibilidad, los controles del proceso y la eficacia también iran avanzando. Existen seis niveles (0 al 5) compuestos de la siguiente manera:


**Gráfico 13. Niveles Arquitectura Empresarial**

**Nivel 0** (No existe Arquitectura Empresarial): En este nivel no existe una planeación para implementar un tipo de Arquitectura Empresarial en la organización. Se carece de documentación sobre tecnologías de información, los procesos no están integrados y varios grupos de empleados se centran en resolver un solo problema al tiempo.

**Nivel 1** (Inicial): Se inicia el desarrollo informal del proceso de Arquitectura Empresarial, realizando un estudio sobre la utilización de un framework existente o se evalúa el desarrollo de una serie de parámetros para implementar un tipo de Arquitectura Empresarial propio. En este nivel se definen algunos procesos, parámetros de documentación y estándares para poder pensar en una unión con los procesos de negocio.

**Nivel 2** (En desarrollo): Se establece el tipo de Arquitectura Empresarial que se utilizará, asimismo los procesos básicos son iniciados, se definen las estrategias, conductores y principios del negocio y la designación métricas de desempeño.

**Nivel 3** (Definida): En este nivel se formalizan los procesos mencionados en el nivel previo, definiendo acertadamente la Arquitectura Empresarial y transmitiéndosela al personal encargado de negocios y tecnologías de información.

**Nivel 4** (Administrada): Se realiza la asociación de métricas de calidad a la Arquitectura Empresarial, midiendo sus procesos. La documentación es actualizada de manera cíclica con el fin de reflejar la actualización de la

arquitectura, en este nivel las arquitecturas de negocios, datos, aplicación y tecnología están completamente definidas.

**Nivel 5 (**Optimizada**):** En este último nivel, convergen todos los anteriores para mejorar los procesos y optimizarlos en base a las necesidades empresariales. Los procesos están en un alto grado de madurez, los objetivos ya han sido determinados de acuerdo a la eficacia y efectividad, por tal motivo se lleva a cabo un refinamiento considerando los cambios y el impacto que estos producen.

Por otro lado, es importante señalar la necesidad de implementar una arquitectura empresarial por parte de las organizaciones, con el fin de obtener una operación más eficiente de TI, un mejor ROI (Return of Investment) y un procurement más rápido, simple y menos costoso, siendo este la principal razón que llevaron a trabajar sobre los procesos de negocios con el objetivo de optimizarlos mediante su integración en busca de la estrategia del negocio.

Muchos autores mencionan, que existen tres razones importantes para utilizar cualquier framework que permita la implementación de una arquitectura empresarial en una organización, a continuación se describen estas tres razones:.

- Permitir la comunicación entre los stakeholders.
- Facilitar la pronta adopción de decisiones de diseño.
- Crear una abstracción transferible de la descripción del sistema.

## 12.3 Características de la Arquitectura Empresarial

La arquitectura empresarial es una descripción de los logros de una organización mediante procesos de negocio atendidos por la tecnología. Se caracteriza por buscar el mejoramiento de los problemas existentes en una organización de manera ordenada, guiándose por estrategias de planeación, buscando siempre una mejora en las actividades para poder adaptarse hacia los nuevos retos y oportunidades que aparecen a diario.

Para llevarse a cabo es necesario un control sobre lo que se ha realizado, lo que se está realizando y lo que se va a realizar dentro de la empresa como afuera de la misma, llevando un manejo de procesos de manera más eficaz y eficiente con una comunicación sólida gracias a modelos de utilidad, esquemas y una narrativa del modo de operación de la organización.

Considerando lo anterior y de acuerdo al "*ChiefInformationOfficer Council*", una Arquitectura Empresarial debe tener:

- **Arquitectura Base:** Donde se especifiquen las prácticas de negocio e infraestructura técnica de la empresa actuales, también se denomina arquitectura "As-Is" o actual.

- **Arquitectura Destino:** Se refleja el pensamiento y los planes estratégicos de la empresa a futuro, denominada también como arquitectura "To-Be".

- **Plan de secuenciación:** Documentación de la transición de la Arquitectura Base a la Destino, contiene actividades, estrategias y desafíos a enfrentar.

## 12.4 Definición de términos relacionados con Arquitectura Empresarial.

A continuación se presentan los siguientes términos definidos en el IEEE Standard 1471-2000 y son fundamentales para el entendimiento de la Arquitectura Empresarial:

- **Arquitecto:** Persona, equipo u organización responsable de la arquitectura.

- **Descripción arquitectónica:** Colección de productos para documentar una arquitectura.

- **Sistema:** Colección de componentes organizados para cumplir una función específica o un conjunto de funciones

- **Stakeholder del sistema:** Individuo, equipo u organización con intereses sobre el sistema.

- **Framework de arquitectura:** Establece los términos y conceptos relacionados con el contenido y el uso de descripciones arquitectónicas.

# 13. FRAMEWORKS DE ARQUITECTURA DE SOFTWARE

## 13.1 Marco de Referencia

Los framework se definen como un conjunto de herramientas de soporte, que proporcionan directrices sobre cómo describir o documentar arquitecturas, permiten la comunicación de los stakeholders de una arquitectura empresarial.

Por otro lado, un marco de referencia sugiere crear:

- Un modelo semántico de las entidades que manipulan las actividades del proceso.
- Un modelo logistico para:
  - Indicar las localidades donde opera el proceso
  - La incorporación de las personas que realizan el trabajo.
  - La identificación de los eventos de negocio que insiden o son caudados por el proceso.
  - La incorporaciòn de las iniciativas estratégicas que se relacionan con el proceso.

De acuerdo al modelo conceptual de la IEEE, señala que: “cada sistema tiene una arquitectura, la cual puede ser registrada en una descripción arquitectónica, y esta solo describe los conceptos de vistas, stakeholders y problemas”. A partir de esta definición se pueden observar los diferentes tipos de frameworks de Arquitectura Empresarial existentes actualmente,considerando las características de cada uno de ellos, se puede seleccionar el más apto para ser implementado en la organización.

La siguiente gráfica se puede apreciar los diferentes frameworks de Arquitectura Empresarial que se han presentado históricamente:[12]

[12] Maritza Mendieta (Febrero 2014), Propuesta de framework de arquitectura empresarial para pymes basado en un análisis comparativo de los framework de zachman togaf, obtenido de: http://dspace.ucuenca.edu.ec/bitstream/123456789/5105/1/TESIS.pdf

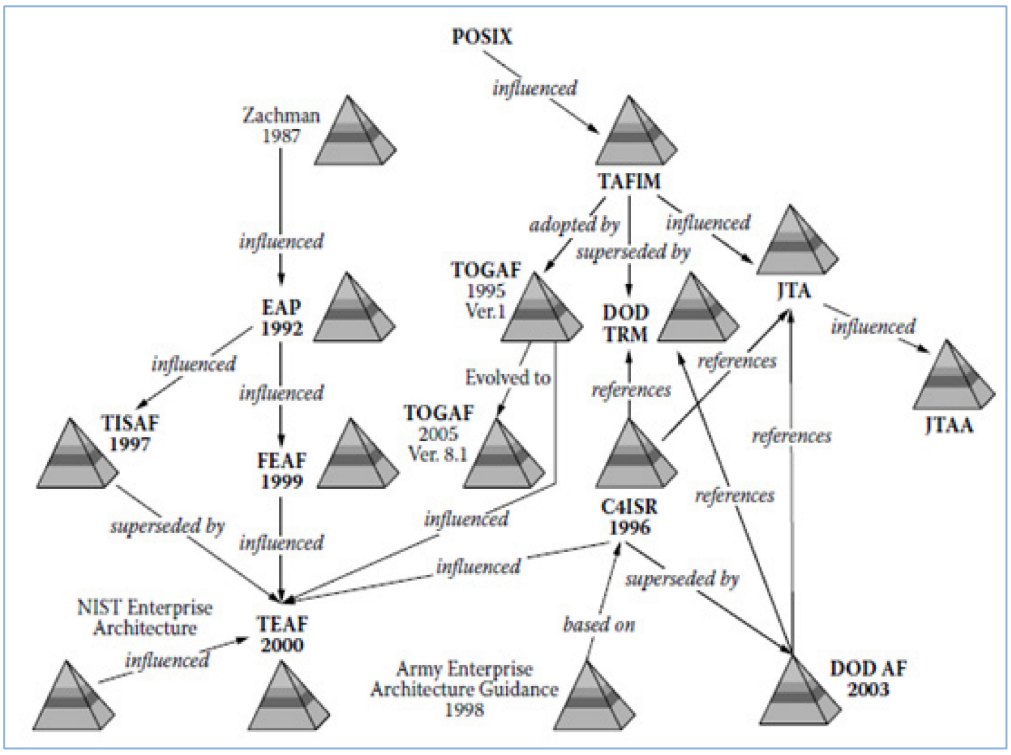



**Gráfico 14. Framework de Arquitectura Empresarial**

Según Roger Sessions, se considera a J.A. Zachman como el precursor de la Arquitectura Empresarial con su artículo “A framework for information systems architecture” publicado en el IBM SystemsJournal en 1987, lo que llevó a la creación del conocido frameworkZachman, todavía vigente siendo este uno de los más utilizados por empresas en estos días.

Como se visualiza en la gráfica anterior, TAFIM (TechnicalArchitecture Framework forInformation Management) fue desarrollado por el Departamento de Defensa de los Estados Unidos como una alternativa para manejar todo el sistema de defensa local desde 1994 y descontinuado 6 años después.

En ese mismo orden de ideas, el TOGAF, el framework más conocido actualmente, ha adoptado en su estructura partes de TAFIM.[13]

[13] Maritza Mendieta (Febrero 2014), Propuesta de framework de arquitectura empresarial para pymes basado en un análisis comparativo de los framework de zachman togaf, obtenido de: http://dspace.ucuenca.edu.ec/bitstream/123456789/5105/1/TESIS.pdf

Como se definió previamente, un framework busca establecer y organizar la información arquitectónica para buscar una evolución sobre los procesos internos de la organización. En la actualidad existen diferentes frameworks, dependiendo de las necesidades de la empresa se puede seleccionar el más apto para ser implementado, a continuación se muestran los más utilizados y sus características generales:

### 13.1.1 Zachman Enterprise Framework

Fue desarrollado por Zachman International, constituido por una tabla de dos dimensiones donde se representa la organización que se analiza. Se encarga de la organización de los artefactos arquitectónicos, por ejemplo, documentos de diseño, especificaciones y modelos, que tengan en cuenta tanto su objetivo como el problema particular a abordar.

Utilizado por diversas empresas, tanto de desarrollo de software como de construcción de edificaciones, hospitales mostrándose muy interesante y para algunos satisfactoria su implementación mediante artículos de interés científico en varias áreas.

Posee un sinfín de beneficios que facilitan el manejo de la información de la empresa, entre los cuales se destacan, facilitar la alineación de TI y el negocio, facilitar la integración de la información a través de los diferentes procesos de negocio, entre otros.

### 13.1.2 Department of Defense Architecture Framework (DoDAF)

Este framework ha sido desarrollado por y con el presupuesto del Departamento de Defensa de Estados Unidos, por lo que cumple con las normas y leyes estipuladas para cualquier entidad del estado y se rige por ellas, además de proporcionar un método para evaluar inversiones, cambios e implementación de tecnologías para cumplir con misiones militares y civiles.

### 13.1.3 Integrated Architecture Framework (IAF)

Fue desarrollado por Capgemini en 1993, bajo la condición de integrar varios tipos de arquitectura con el framework y de la misma manera unir el vocabulario de las diferentes comunidades.

El manejo de información es muy parecido al de Zachman, pero en vez de enfocarse en seis preguntas principales, lo realiza en cuatro:

- ¿Por qué?
- ¿Qué?
- ¿Cómo?
- ¿Con qué?

Se adapta fácilmente a las necesidades del usuario y es escalable desde proyectos individuales a los que integran a toda una organización, asimismo es reconocida e implementada en varias empresas a nivel mundial.[14]

### 13.1.4 The Open Group Architecture Framework (TOGAF)

Existen más frameworks de Arquitectura Empresarial, pero esta es considerada una de las más importante.

Desarrollado por The Open Group, este framework define a la empresa como: "cualquier colección de organizaciones con unos objetivos en común" por lo que busca "proveer los métodos y herramientas para asistir en la aceptación, producción, uso y mantenimiento de una arquitectura empresarial, basado en un modelo de procesos iterativo soportado por buenas prácticas y un conjunto reutilizable de los activos de la arquitectura existente".[15]

#### 13.1.4.1 Estructura de TOGAF

La parte más importante de este *framework* está compuesto por su "Architecture Development Method" o ADM. Siendo este el que define el proceso a realizar en la arquitectura, llevando lo más genérico a lo más específico. Este ADM está conformado por las siguientes fases y se organiza de forma iterativa y cíclica de la siguiente manera:

[14] José Martinez & Camilo Silva, (2010), Guia metodológica para el levantamiento y análisis de requerimientos de software con base en procesos de negocios. Obtenido de: http://pegasus.javeriana.edu.co/~CIS1010IS06/Entregables.html

[15] José Martinez & Camilo Silva, (2010), Guia metodológica para el levantamiento y análisis de requerimientos de software con base en procesos de negocios. Obtenido de: http://pegasus.javeriana.edu.co/~CIS1010IS06/Entregables.html

**La fase A**, denominada visión de la arquitectura, define los límites que permitirán medir el alcance del proyecto y la estrategia para lograrla. Se ejecuta con el fin de validar el contexto del negocio y producir una declaración del trabajo de arquitectura aprobada.

**La fase B**, llamada arquitectura del negocio, busca tener clara la arquitectura del negocio y las metas que quiere cumplir para revisar si es viable o no complementarla con TI. Es decir, aborda el desarrollo de una arquitectura de negocio que apoye la visión de la arquitectura acordada.

**La fase C**, nombrada arquitectura de sistemas de información contempla las arquitecturas particulares para datos y aplicaciones, estos pueden ser desarrollados simultáneamente o de manera secuencial.

**La fase D**, denominada arquitectura tecnológica, define la arquitectura integrada para el desarrollo de las fases posteriores, en esta fase se aborda la documentación de la organización esencial de sistemas de TI, representada en hardware, software y tecnología de la información.

**La fase E**, llamada oportunidades y soluciones, permite determinar un inventario de elementos con los cuales se cuentan para montar la fase D. En ella se determina cuáles componentes es necesario comprar, modificar o arreglar para que pueda ser útil en la arquitectura. Es la primera fase que se refiere directamente a la implementación.

**La fase F**, nombrada plan de migración, prioriza los proyectos paralelos y gestiona un plan de migración de la empresa al sistema construido.

**La fase G**, denominada control de la implementación, es la ejecución de los proyectos para construir las soluciones de TI.

**La fase H**, llamada administración del cambio de la arquitectura, monitorea y evalúa los sistemas existentes para determinar cuándo iniciar un nuevo ciclo de ADM, en esta fase se debe asegurar que los cambios en la arquitectura se gestionen de una manera controlada.

#### 13.1.4.2 Dominios de Arquitectura Empresarial y TOGAF

Este Framework, está diseñado para soportar los cuatro dominios que son reconocidos como parte de una arquitectura empresarial, estos son descritos de la siguiente manera por Open Group:

- **Business** (Negocio): En este se encuentran incluidos la estrategia del negocio, procesos clave del negocio, entre otros.
- **Data** (Datos): Recursos sobre el manejo de datos lógicos y físicos.
- **Application** (Aplicación): Es una descripción de las capacidades para administrar los datos existentes.
- **Technology** (Tecnología): El software y hardware capaz de soportar los servicios de negocio, datos y aplicación.

El más importante sin lugar a dudas es el de negocios, ya que el objetivo principal es suplir todas las necesidades a través de sus procesos, asignándoles funcionalidades con el fin de realizar un diagrama de procesos.[16]

### 13.1.5 Framework ATOM

Atom es un Framework para el desarrollo de arquitectura empresarial, sus siglas significan: Arquitectónico, tecnológico, organización y gestión, este framework trabaja en base a la visión y misión de la emprersa y sobre lo que debe trabajar para el logro de los objetivos, no quiere decir que sea una afirmación de cómo crear beneficios, sin embargo con ello se puede sembrar valores.

Por otro lado, es importante definir claramente la estrategia de la empresa y desde alli se pueden ir identificando las sub-estrategias como por ejemplo, la estrategia financiera, la estrategia de recursos humanos (planificación de la plantilla), la estrategia de comunicación y planificación de la tecnología.

Sin embargo existe una sub especial - estrategia que se diferencia del resto de las sub - estrategias y es la denominada estrategia de TI y la

[16] Andrew Josey, Rachel Harrison, Paul Homan, Matthew Rouse, Tom Sante, Mike Turner, Paul Van Der Merwe, (2013), Togaf Version 9.1 Guia de bolsillo.

sección de gobierno de TI. Todo ello debido a la importancia de la tecnología de la información en el transcurso de los años.

A continuación se describen los apartados del Framework ATOM:

- **Arquitectura**

  La teoria es que cada empresa tiene una arquitectura siendo esta una necesidad para que la empresa pueda realizar las actividades que crean valor. Cada empresa tiene una arquitectura sino cómo la arquitectura empresarial puede ayudar a la empresa a obtener una ventaja competitiva. Para ello, la arquitectura de la empresa debe ir madurando en el transcurso del tiempo..

  Una vez que haya madurado existen varios puntos que el equipo ejecutivo tiene que hacer frente a la arquitectura para lograr mejores resultados que con el tiempo permitan ayudar a la empresa lograr ventajas competitivas.

- **Tecnología**

  La hipótesis principal es que la tecnología es una herramienta que se puede utilizar para lograr mejores resultados para la empresa, por ejemplo, computadoras, correos electrónicos y sistemas de información. Por otro lado, la empresa debe considerar como tecnologia autos, máquinas y otras cosas que se usan para hacer que los empleados, gerentes y altos directivos logren las metas y visiones de la empresa.

- **Organización**

  La empresa consta de personas. La gente tiene que cambiar la forma en que hacen su trabajo, interactuan con otros y su manera de pensar cuando trabajan.

- **Gerencial**

  El aspecto de gestión o gerencial del framework debe estar alineado con el equipo ejecutivo de ventas de la empresa y la toma de decisiones, lo cual es necesario para aplicar los cambios necesarios que permitan a la empresa la consecución de sus objetivos.

- **Estructura del Framework ATOM**

  La fase inicial del framework se basa en la idea de que la arquitectura es la fuerza motriz, entonces la tecnología permite a voluntad, la organización es construir sobre el ajuste de la conducta de los miembros de la empresa, como los directivos, mandos intermedios, empleados, entre otros.

| Arquitectura | Tecnología | Organización | Gerencial |
|---|---|---|---|
| | | | |

Tabla 3 Framework ATOM

Al aplicar el framework, es necesario pensar en el framework y el concepto de arquitectura de la empresa con el apoyo de la gestión (gerencial), lo cual genera una reorganización, colocando este en segundo lugar, dado que la tecnología es un facilitador para lograr la ventaja competitiva, no debe ser considerada como secundaria.

Entonces ¿por qué es la arquitectura en frente de la organización y el nivel gerencial? La razón de esto es que todas las empresas tienen una arquitectura y cuando la arquitectura madura, la empresa es capaz de lograr mejores resultados a partir de sus elementos administrativos, organizativos y tecnológicos.

| Arquitectura | Gerencial | Organización | Tecnología |
|---|---|---|---|
| | | | |

Tabla 4 Framework ATOM – Reorganizado

El framework ATOM se puede suponer que la empresa de alguna manera se organiza como una antigua pirámide egipcia.

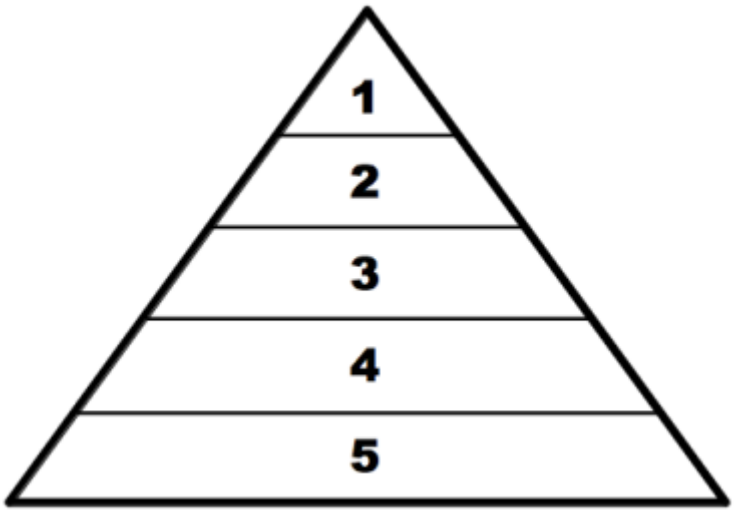



**Gráfico 15. Pirámide de la Organización**

El **primer** nivel se refiere a la gestión de la empresa y la formulación de la estrategia corporativa.

El **segundo** nivel se refiere a los modelos de negocio y procesos de negocio de la empresa. Esto describe cómo la empresa genera valor a sus clientes.

El **tercer** nivel se ocupa del negocio de TI fase de alineación. Esto significa que la empresa se centra en hacer su trabajo como es debido.

El **cuarto** nivel tiene que ver con los artefactos de información relacionada, por ejemplo, cómo son los sistemas de información y bases de datos diseñadas.

El **quinto** nivel parte de la idea de que cada otra capa en la empresa se relaciona o construye sobre el uso de Tecnologías de la Información.

### 13.1.6 Evaluación y comparación de los principales framework de arquitectura de software empresarial

En la siguiente tabla se muestra una breve comparación de los principales frameworks de arquitectura de software empresarial, los criterios considerados en la misma esta estrechamente relacionado con los beneficios que obtendrá la empresa luego de la implementación de una arquitectura empresarial, como lo son:

- Retorno de la inversión.
- Reingeniería de procesos.
- Valor agregado a los servicios y a los productos que existen.
- Generación de nuevos servicios y/o productos.

| Criterio | Zachman | Togaf | Fea | Gartner | Atom |
|---|---|---|---|---|---|
| Integridad taxonómica | 4 | 2 | 2 | 1 | 2 |
| Integridad del proceso | 1 | 4 | 2 | 3 | 4 |
| Modelo de referencia | 1 | 3 | 4 | 1 | 3 |
| Orientación práctica | 1 | 2 | 2 | 4 | 1 |
| Modelos de madurez | 1 | 1 | 3 | 2 | 1 |
| Enfoque de negocio | 1 | 2 | 1 | 4 | 4 |
| Orientación a gobernanza | 1 | 2 | 3 | 3 | 3 |
| Orientación a partición | 1 | 2 | 4 | 3 | 2 |
| Catálogo prescriptivo | 1 | 2 | 4 | 2 | 1 |
| Neutralidad de proveedores | 2 | 4 | 3 | 1 | 2 |
| Disponibilidad de la información | 2 | 4 | 2 | 1 | 1 |
| Tiempo de valoración | 1 | 3 | 1 | 4 | 3 |

**Gráfico 16. Comparación de framework**

| Escala | Significado | Definición |
|---|---|---|
| **1** | Deficiente | El desempeño del Framework, se desempeña muy pobremente. |
| **2** | Insuficiente | El desempeño del Framework, se desempeña inadecuadamente. |
| **3** | Aceptable | El desempeño del Framework es bueno pero puede mejorar. |
| **4** | Satisfactorio | El desempeño del Framework, es muy bueno en esta área. |

Tabla 9 Escala de criterios a evaluar

**Integridad taxonómica** es el grado en el que se puede utilizar el framework para clasificar los distintos artefactos de la arquitectura.

**Integridad del proceso** se refiere a como la metodología guía paso a paso a través de un proceso para la creación de una arquitectura empresarial.

**Modelo de referencia** es la utilidad de la metodología y cuanto ayuda a construir un modelo relevante de referencia y cuan prácticos son los modelos presentados por el framework para construir nuevos.

**Orientación práctica** se refiere a cuanto ayuda el framework a que la organización asimile la mentalidad de implementar los artefactos de la arquitectura y a la vez desarrollar una cultura en donde se valore y utilice constantemente los componentes del framework.

**Modelos de madurez** se refiere a la cantidad de orientación que ofrece la metodología para que se dé una evaluación eficaz y una correcta madurez de las diferentes organizaciones mediante el uso de una arquitectura empresarial.

**Enfoque de negocio** se refiere a si la metodología se centra en el uso de la tecnología para impulsar el valor del negocio, en el que esté definido como la reducción de gastos y/o aumento de ingresos.

**Orientación a gobernanza** se refiere a la cantidad de ayuda que la metodología ofrece en la comprensión y la creación de un modelo de gestión eficaz para la arquitectura empresarial.

**Orientación a partición** se refiere a lo bien que la metodología guiara en el particionamiento de empresas autónomas, que es un enfoque importante para gestionar la complejidad.

**Catálogo prescriptivo** se refiere a que tan bien la metodología guía en la creación de un catálogo o un patrimonio arquitectónico que puede ser reutilizado en futuras actividades.

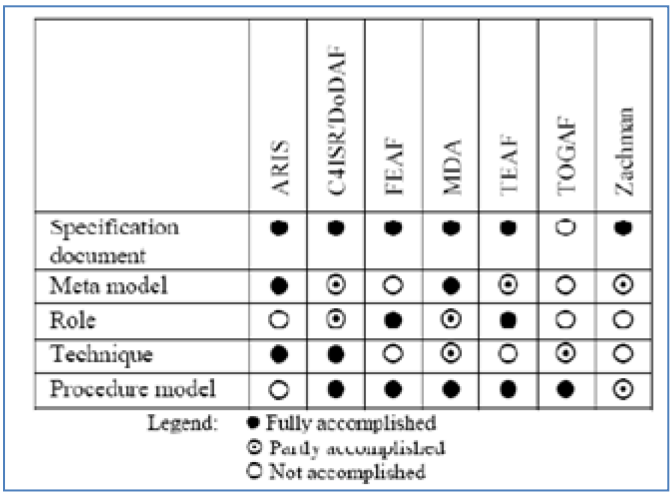
**Consideración de factores externos / neutralidad de proveedores** se refiere a cuan flexibles son los modelos del framework de ser modificados ante los agentes de cambio externos del mercado (por ejemplo: cambios regulatorios en la industria, nuevas innovaciones en productos y servicios, etc.)

**Disponibilidad de la información** se refiere a la cantidad y calidad de información gratuita o de bajo costo de esta metodología. Es el grado en el cual el framework ha llegado a evolucionar en la medida de que otras empresas hayan desarrollado herramientas que permitan estimar, planear y administrar las distintas fases y artefactos de la arquitectura.

**Tiempo de valoración** se refiere a la cantidad de tiempo que probablemente el framework consumirá antes de empezar a construir soluciones que proporcionen un alto valor agregado.

### 13.1.7 Evaluacion de frameworks de acuerdo al estudio de leist y zellner

En el siguiente gráfico se muestra un comparativo de siete Frameworks con tres calificaciones diferentes de acuerdo a cinco criterios específicos según el estudio realizado por Leist y Zellner:



| | ARIS | C4ISR/DoDAF | FEAF | MDA | TEAF | TOGAF | Zachman |
|---|---|---|---|---|---|---|---|
| Specification document | ● | ● | ● | ● | ● | ○ | ● |
| Meta model | ● | ⊙ | ○ | ● | ⊙ | ○ | ⊙ |
| Role | ○ | ⊙ | ● | ⊙ | ● | ○ | ○ |
| Technique | ● | ● | ○ | ⊙ | ○ | ⊙ | ○ |
| Procedure model | ○ | ● | ● | ● | ● | ● | ⊙ |

Legend: ● Fully accomplished
⊙ Partly accomplished
○ Not accomplished

**Gráfico 17. Evaluación de Frameworks (Leist y Zellne)**

- Tabla de Calificación

| Escala | Significado |
|---|---|
| ● | Cumple Plenamente |
| ⊙ | Cumple en parte |
| ○ | No cumple |

Tabla 10 de calificación de acuerdo al estudio

Cada framework tiene fortalezas y debilidades, los frameworks no cumplen con todos los requisitos relativos a los elementos constitutivos de un método, sin embargo siempre existirá uno que se acople a las necesidades de la empresas, representadas estas en sus estrategias financieras, de negocios y finalmente tecnologicas.

La determinación de conocer cual es el rumbo de la empresa estara siempre fijada en los objetivos que corto, mediando y largo plazo se establezcan para cumplir con la misión y visión de la organización.

## CONCLUSIONES

La motivación de este trabajo nace de la necesidad imperante de conocer sobre un punto de gran importancia en el ámbito de la informática como es la arquitectura de software, a medida de su desarrollo se fue develando cada uno de los tópico motivo de esta investigación, si se considera el hecho o la realidad de las empresas u organizaciones en la actualidad, donde todas ellas dependan de un software (sistema empresarial) que le permita realizar la gestión empresarial. Sin embargo para llegar a este software, el recorrido fue largo.

Todo comienza en la definición de una arquitectura que le permita englobar de una forma significante todos los requerimiento y ademas se encuentre enfocado a las reglas propias del negocio, es por ello que el punto inicial en un proyecto es la definición de la arquitectura que se empleara, uno de los objetivos que se busca con ello, es obtener como producto una aplicación empresarial que le facilite la gestión con los clientes pero a su vez le brinde la posibilidad de optimizar sus recursos internos y satisfacer las necesidades del negocio.

En este sentido es importante resaltar la importancia de especificar los requerimientos y realizar el modelado integrado que proporcione los métodos y la tecnología que se requerira para el diseño e implementación de la arquitectura corporativa adecuada, para llegar a ello se debe definir claramente el diseño, el alcance, asi como también el tipo de framework a utilizar entre otros aspectos de vital importancia.

El éxito de un desarrollo optimo de una aplicación empresarial depende de la disposición de cada miembro del equipo encargado de la planeación, diseño, desarrollo e implementación de la misma, esta investigación sirve de referencia para la evaluación de los diferentes estilos arquitectonicos que pueden ser utilizados en los desarrollos empresariales.

Si se considera todo lo analizado en la presente investigación, se puede determinar que una arquitectura empresarial es fundamental para lograr que TI soporte y facilite los procesos de negocio de una organización, ya que permite alinear la estrategia de negocio con la infraestructura de comunicación y los servicios de información de una empresa.

Una arquitectura empresarial debe brindar valor real al negocio lo más rápido posible. Uno de los objetivos más importantes de cualquier arquitectura

empresarial es llevar la parte comercial y la tecnológica de la mano para que ambas estén trabajando de manera efectiva hacia los mismos objetivos.

En este sentido se puede decir que una arquitectura empresarial garantiza que los requisitos de la organización se cumplan, a través de la integración de una estrategia de tecnologías de la información, permitiendo la mayor concordancia posible en los procesos que maneja la organización.

Por otro parte, es importante seleccionar un framework apropiado para lograr una descripción clara y completa de una arquitectura empresarial, que contribuya a que la organización cumpla sus fines y metas.

Cada framework es un creación unica, el cual fue creado para satisfacer una necesidad de una industria específica, por ello se debe analizar detenidamente antes de tomar una decisión sobre su uso, al elegir el framework se debe analizar la estructura de la empresa al igual que sus procesos.

Implementar sabiamente un framework es una decisión crítica, tanto de negocios como de estrategia, con esto se asegura de no ignorar ningún aspecto de la organización.

Por otro lado, considerando todos los aspectos documentados en la presente investigación, lo esencial es saber combinar todos los ejes que mueven a una arquitectura empresaria y por ende el seleccionar el framework adecuado va a determinar el éxito de la solución que se quiere alcanzar.

Es por ello que la arquitectura de software por si sola no daria los frutos que se quiere, si no se cuenta con una buena gerencia dispuesta a llevar a cabo todos los aspectos que implican una arquitectura empresarial, pero tambien es importante comunicar en todo momento y en todos los niveles de la organización los cambios que se estan realizando de tal manera que no existe una resistencia al cambio que pudieran obstaculizar el logro y éxito de la implementación de la arquitectura empresarial.

Finalmente se puede concluir, que esta investigación proporcionará una guía inicial al lector, sobre el uso de frameworks de arquitectura de software en proyectos de ingeniería de software empresarial. A partir de los conocimientos adquiridos en esta investigación, el arquitecto de software o ingeniero de sotware, podrá complementar sus conocimientos y aplicar técnicas y procedimientos descritos en esta investigación. Esta investigación promueve una invitación al lector, a profundizar en los temas aquí descritos y que apliquen

al entorno laboral en el cual se desempeñe, para obtener mejores técnicas, procedimientos y prestaciones en el desarrollo de los proyectos.

## BIBLIOGRAFIA


- (informática), W. -A. (21 de 06 de 2014). Obtenido de http://es.wikipedia.org/wiki/Arquitectura_en_pipeline_(inform%C3%A1tica)
- Abraham Sanchez Juárez - Arquitectura por capas, c.-s. T. (s.f.). http://www.joomag.com/. Obtenido de http://www.joomag.com/magazine/evidencia-ii-arquitectura-de-software-vol1/0343642001389052240?page=5
- Bas, J. P. (29 de 05 de 2009). Blog de Juan Peláez en Geeks.ms. Recuperado el Marzo de 2015, de http://geeks.ms/blogs/jkpelaez/archive/2009/05/29/arquitectura-basada-en-capas.aspx
- BASS Len, C. P. (1998). Software Architecture in Practice.
- Bass, P. C. (2003). “Software Architecture in Practice”.
- Cervantes, H. (28 de 05 de 2010). SG BUZZ. Recuperado el Marzo de 2015, de http://sg.com.mx/revista/28/requerimientos-y-arquitectura#.VQnFlCuUcWh
- FJC, G. d. (s.f.). http://es.slideshare.net/glud/conceptos-basicos-arquitectura-de-software.
- GARLAN, D. Next generation software architectures: Recent research and future directions.
- GARLAN, D. y. An introduction to software architecture. CMU Software Engineering.
- Goethals, F. (2003). An overview of enterprise architecture framework deliverables. DTEW Research Report0570.
- Greta, J. .. (25 Octuber 2005). Enterprise Architecture Framework: Evolution 2005 Research No. G00130855.
- Group, T. O. (16 de 10 de 2012). Other Architectures and Architectural Frameworks. Obtenido de http://www.opengroup.org/public/arch/p4/others/others.htm#TAFIM

- Group, T. o. (08 de 02 de 2011). The open group (Merking standards work) . Recuperado el Marzo de 2015, de http://www.opengroup.org/subjectareas/enterprise/togaf/
- Guglielmetti, M. (s.f.). http://www.mastermagazine.info/termino/3916.php .
- Información., U. S. (15 de 03 de 2011). http://ldc.usb.ve/ (Laboratorio docente de computación). Obtenido de http://ldc.usb.ve/~teruel/ci3715/clases/arqCapas2.html.
- Miller, J. M. (2001). Model Driven Architecture. Technology Committee and Architecture Boar.
- pizarra, W. -A. (14 de 3 de 2013). Wikipedia.org. Obtenido de http://es.wikipedia.org/wiki/Arquitectura_en_pizarra_(inform%C3%A1tica)
- Recursos, D. R.-A. (s.f.). Dairelysrodriguez Wikia. Recuperado el Marzo de 2015, de http://es.dairelysprogramacion.wikia.com/wiki/Arquitectura_Orientada_a_Recursos_(AOR)
- Repositorio. (2 de 2 de 2015). http://danimaniarqsoft.com/. Recuperado el Marzo de 2015, de http://danimaniarqsoft.com/?p=182
- Scheer, A. (1992). Architecture for Integrated Information Systems, Berlin: SpringerVerlag.
- servicios, A. o. (19 de Octubre de 2011). http://soa-fpuna.blogspot.com/. Recuperado el Marzo de 2015, de http://soa-fpuna.blogspot.com/2011/11/ventajas-y-desventajas.html
- Shaw, D. G. (1994). An Introduction to Software Architecture Publishing Company. Pittsburgh, PA : Ambriola and G.Tortora, World Scientific.
- software, V. V. (18 de 06 de 2008). http://www.mailxmail.com/. Obtenido de http://www.mailxmail.com/curso-programacion-avanzada/frameworks-arquitecturas-software
- Standarization, I. O. (2006). Framework for Enterprise Modelling. EN/ISO 19439. Geneva: ISO.
- Systems, 1.-2. -I.-I. (s.f.). standards.ieee.org/. Recuperado el Marzo de 2015, de http://standards.ieee.org/findstds/standard/1471-2000.html

- Tang, A., Han, J., & Chen, P. (2004). A Comparative Analysis of Architecture Frameworks. Technical Report SUTIT-TR2004.01. School of Information Technology of Swuinburne.
- Urbaczewski, L. y. (2006). A comparison of enterprise architecture frameworks. Issues in Information Systems. Vol. VII, No. 2.
- Venezuela, D. G. (Mayo de 2011). http://www.codecompiling.net/files/slides/IS_clase_08_estilos_arquitectonicos.pdf.
- Wikipedia.org. (Marzo de 2015). Obtenido de http://es.wikipedia.org/wiki/Cliente-servidor
- Zachman, J. (1987). A framework for information systems architecture. IBM Systems Journal, Vol 26, No 3,.