



EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA



TECNOLÓGICO
NACIONAL DE MÉXICO®



Especialidad Análisis, diseño y desarrollo de software

**D3.1 Estilos, patrones y diseño arquitectónico de
software**

Asesor: M.T.I.C. Leonardo Enriquez

Ingeniero electrónico, sistemas digitales

Arquitectura de Software

La arquitectura de software de un programa o sistema de computo es la estructura o estructuras del sistema, lo que comprende a los **componentes del software, sus propiedades externas visibles y las relaciones entre ellos**.


La arquitectura de software es una representación que permite

1. Analizar la efectividad del diseño para cumplir los requerimientos establecidos.
2. Considerar alternativas arquitectónicas en una etapa e la que hacer cambios al diseño todavía es relativamente fácil.
3. Y reducir los riesgos asociados con la construcción del software.

La definición anterior pone el énfasis en el papel de los “**componentes del software**” en cualquier representación arquitectónica. En el contexto del diseño de la arquitectura, un componente de software puede ser algo tan simple como un modulo de programa o una clase orientada a objeto, pero también puede incluir base de datos y “middleware” que permitan la configuración de un red de clientes y servidores.

Un diseño es una instancia de una arquitectura, similar a un objeto que es una instancia de una clase. Por ejemplo, considera la arquitectura cliente-servidor. Con esta arquitectura es posible diseñar de muchos modos un sistema de software basado en red, con el uso de una plataforma Java o Microsoft, entonces hay una arquitectura pero con base en ella pueden crear muchos diseños, así, **no es valido mezclar arquitectura y diseño**.

IEE Computer Society define una descripción arquitectónica como un conjunto de productos para documentar una arquitectura. La descripción en si se representa con el uso de perspectivas múltiples, donde cada **perspectiva es una representación del sistema** completo desde el punto de vista de un conjunto de preocupaciones relacionadas.



Formato para la descripción de una decisión arquitectónica

Toda decisión arquitectónica de importancia puede documentarse para que posteriormente la revisen los participantes que deseen entender la descripción de la arquitectura propuesta. El formato que se presenta en este recuadro es una versión adaptada y abreviada de otro, propuesto por Tyree y Ackerman [Tyr05].

Aspecto del diseño:	Se describen los aspectos del diseño arquitectónico que se van a abordar.	Alternativas:	Se describen con brevedad las alternativas de diseño arquitectónico que se consideraron y la razón por la que se rechazaron.
Resolución:	Se establece el enfoque escogido para abordar el aspecto de diseño.	Argumento:	Se establece por qué se eligió la resolución sobre las demás alternativas.
Categoría:	Se especifica la categoría de diseño a que se aboca el aspecto y la resolución (por ejemplo, diseño de datos, estructura del contenido, estructura del componente, integración, presentación, etcétera).	Implicaciones:	Se indican las consecuencias que tendrá la toma de la decisión en el diseño. ¿Cómo afectará la resolución a otros aspectos del diseño de la arquitectura? ¿La resolución restringe de algún modo al diseño?
Suposiciones:	Se indican cualesquiera suposiciones que ayuden a dar forma a la decisión.	Decisiones relacionadas:	¿Qué otros documentos se relacionan con esta decisión?
Restricciones:	Se especifican todas las restricciones ambientales que ayuden a conformar la decisión (como los estándares tecnológicos, patrones disponibles y aspectos relacionados con el diseño).	Preocupaciones relacionadas:	¿Qué otros requerimientos se relacionan con esta decisión?
		Productos finales:	Se indica dónde se reflejará esta decisión en la descripción arquitectónica.
		Notas:	Se hace referencia a las anotaciones del equipo u otra clase de documentación que se haya empleado para tomar la decisión.

Estilos arquitectónicos a un genero especifico (dominio de aplicación)

Aunque los principios del diseño arquitectónico se aplican a todos los tipos de arquitectura, con frecuencia será el genero arquitectónico el que dice el enfoque especifica para la estructura que deba considerarse. En el contexto del diseño arquitectónico, el genérico implica una categoría especifica dentro del dominio general del software. Dentro de cada categoría hay varias subcategorías, y dentro de cada estilo habrá estilos mas especifico, por lo que cada estilo tendrá una estructura que puede describirse con el uso de un conjunto de patrones predecibles.

Grady Booch en su texto handbook of software architecture, sugiere entre otros los siguientes géneros arquitectónicos para sistemas basados en software:

- **Inteligencia artificial:** Sistemas que simulan o incrementan la cognición humana, su locomoción u otros procesos orgánicos.
- **Comerciales y no lucrativos:** Sistemas que son fundamentales para la operación de una empresa de negocios.
- **Comunicaciones:** Sistemas que proveen la infraestructura para transferir y manejar datos, para conectar usuarios de éstos o para presentar datos en la frontera de una infraestructura.
- **Dispositivos:** Sistemas que interactúan con el mundo físico a fin de brindar algún servicio puntual a un individuo.
- **Juegos:** Sistemas que dan una experiencia de entretenimiento a individuos o grupos.
- **Industrial:** Sistemas que simulan o controlan procesos físicos.
- **Médicos:** Sistemas que diagnostican, curan o contribuyen a la investigación médica.
- **Utilidades:** Sistemas que interactúan con otro software para brindar algún servicio específico.

Ejemplo de un estilos arquitectónico a un sistema de juego

Desde el punto de vista del diseño arquitectónico cada genero representa un desafío, por ejemplo considere la **arquitectura del software de un sistema de juego**. Esta clase de sistemas, en ocasionados llamados aplicaciones interactivas de inmersión, requieren el computo de algoritmos intensivos, graficas avanzadas en computadora, fuente de datos continuas en multimedios, interactividad en tiempo real a través de dispositivo de entrada convencionales y no convencionales, y otras preocupaciones especializadas.

La ASI (Arquitectura de Software de Innerpresencia) es un modelo nuevo de **arquitectura de software** para diseñar, analizar e implementar aplicaciones que realizan un **procesamiento distribuido, asíncrono y paralelo de flujos de datos** generales. El objetivo de la ASI es proveer un marco universal para la implementación distribuida de algoritmos y su fácil integración en sistemas complejos. El modelo de procesamiento de datos extensibles subyacentes y el modelo de procesamiento **híbridos, asíncrono y en paralelo** permiten la manipulación natural y eficiente de flujos de datos generales con el uso indistinto de **librerías** ya existentes o código original. La modularidad del estilo facilita el desarrollo de código distribuido, pruebas y reutilización, así como el diseño e integración rápida del sistema y su mantenimiento y evolución.

Estilo arquitectónico

El software construido para sistemas basado en computadoras tiene uno de muchos estilos arquitectónicos. Cada **estilo describe una categoría de sistemas** que incluye:

1. Un conjunto de **componentes** (como una base de datos o módulos de computo) que realizan una función requerida por el sistema.
2. Un conjunto de **conectores** que permiten la comunicación, coordinación y cooperación entre los componentes
3. **Restricciones** que define como se integran los componentes para formar el sistema.
4. Modelos **semánticos** que permiten que un diseñador entienda las propiedades generales del sistema al analizar las propiedad conocidas de sus partes constituyentes.



Estructuras arquitectónicas canónicas

En esencia, la arquitectura del software representa una estructura en la que cierta colección de entidades (con frecuencia llamados *componentes*) está conectada por un conjunto de relaciones definidas (usualmente llamadas *conectores*). Tanto las componentes como los conectores están asociados con un conjunto de *propiedades* que permiten que el diseñador diferencie los tipos de componentes y conectores que pueden usarse. Pero, ¿qué clases de estructuras (componentes, conectores y propiedades) se utilizan para describir una arquitectura? Bass y Kazman [Bas03] sugieren cinco estructuras canónicas o fundamentales:

Estructura funcional. Los componentes representan entidades de función o procesamiento. Los conectores representan interfaces que proveen la capacidad de “usar” o “pasar datos a” un componente. Las propiedades describen la naturaleza de los componentes y la organización de las interfaces.

Estructura de implementación. “Los componentes son paquetes, clases, objetos, procedimientos, funciones, métodos, etc., que son vehículos para empaquetar funciones en varios niveles de abstracción” [Bas03]. Los conectores incluyen la capacidad de pasar datos y control, compartir datos, “usar” y “ser una instancia de”. Las propiedades se centran en las características de la calidad (por ejemplo, facili-

dad de recibir mantenimiento, ser reutilizables, etc.) que surgen cuando se implementa la estructura.

Estructura de concurrencia. Los componentes representan “unidades de concurrencia” que están organizadas como tareas o trayectorias paralelas. “Las relaciones [conectores] incluyen sincronizarse-con, tiene-mayor-prioridad-que, envía-datos-a, no-corre-sin y no-corre-con. Las propiedades relevantes para esta estructura incluyen prioridad, anticipación y tiempo de ejecución” [Bas03].

Estructura física. Esta estructura es similar al modelo de despliegue desarrollado como parte del diseño. Los componentes son el hardware físico en el que reside el software. Los conectores son las interfaces entre los componentes del hardware y las propiedades incluyen la capacidad, ancho de banda y rendimiento, entre otros atributos.

Estructura de desarrollo. Esta estructura define los componentes, productos del trabajo y otras fuentes de información que se requieren a medida que avanza la ingeniería de software. Los conectores representan las relaciones entre los productos del trabajo; las propiedades identifican las características de cada aspecto.

Cada una de estas estructuras presenta un punto de vista de la arquitectura del software y expone información que es útil para el equipo de software a medida que realiza la modelación y construcción.

INFORMACIÓN

Estilo arquitectónico

Shaw y Garlan (1996) define estilo arquitectónico como una familia de sistemas en términos de un patrón de organización estructural, que define un vocabulario de componentes y tipos de conectores y un conjunto de restricciones de como pueden ser combinadas.

La tabla resume los principales **estilo arquitectónicos**, **los atributos de calidad** que propician y los atributos que se ven afectados negativamente (atributos de conflicto), de acuerdo a Bass et al. (1998)

Un **estilo arquitectónico basado en atributos** incluye:

- La topología de los tipos de componentes y una descripción del patrón de los datos y control de interacción entre ellos, de acuerdo con la definición estándar.
- Un modelo específico de atributos de calidad que provee un método de razonamiento acerca del comportamiento de los tipos de componentes que interactúan en el patrón definido.
- El razonamiento que resulta de la aplicación del modelo específico de atributos de calidad a la interpretación de los tipos de componentes.

Un estilo arquitectónico basado en atributos **consta de cinco partes** (Bass et al., 1999) que se muestran en la tabla.

Elemento	Descripción
Descripción del problema	Describe el problema de diseño que el ABAS pretende resolver, incluyendo el atributo de calidad de interés, el contexto de uso, y requerimientos específicos relevantes al atributo de calidad asociado
Medidas del atributo de calidad	Contiene los aspectos medibles del modelo de atributos de calidad. Incluye una discusión de los eventos que causan que la arquitectura responda o cambie
Estilo Arquitectónico	Descripción del estilo arquitectónico en términos de componentes, conectores, propiedades de los componentes y conexiones, así como patrones de datos y control de interacciones
Parámetros de atributos de calidad	Especificación del estilo arquitectónico en términos de los parámetros del modelo de calidad
Análisis	Descripción de cómo los modelos de atributos de calidad están formalmente relacionados con los elementos del estilo arquitectónico y las conclusiones acerca del comportamiento arquitectónico que se desprende de los modelos

Estilos arquitectónico, atributos asociados y en conflicto

Estilo	Descripción	Atributos asociados	Atributos en conflicto
<i>Datos Centralizados</i>	Sistemas en los cuales cierto número de clientes accede y actualiza datos compartidos de un repositorio de manera frecuente.	Integrabilidad Escalabilidad Modificabilidad	Desempeño
<i>Flujo de Datos</i>	El sistema es visto como una serie de transformaciones sobre piezas sucesivas de datos de entrada. El dato ingresa en el sistema, y fluye entre los componentes, de uno en uno, hasta que se le asigne un destino final (salida o repositorio).	Reusabilidad Modificabilidad Mantenibilidad	Desempeño
<i>Máquinas Virtuales</i>	Simulan alguna funcionalidad que no es nativa al hardware o software sobre el que está implementado.	Portabilidad	Desempeño
<i>Llamada y Retorno</i>	El sistema se constituye de un programa principal que tiene el control del sistema y varios subprogramas que se comunican con éste mediante el uso de llamadas.	Modificabilidad Escalabilidad Desempeño	Mantenibilidad Desempeño
<i>Componentes Independientes</i>	Consiste en un número de procesos u objetos independientes que se comunican a través de mensajes.	Modificabilidad Escalabilidad	Desempeño Integrabilidad

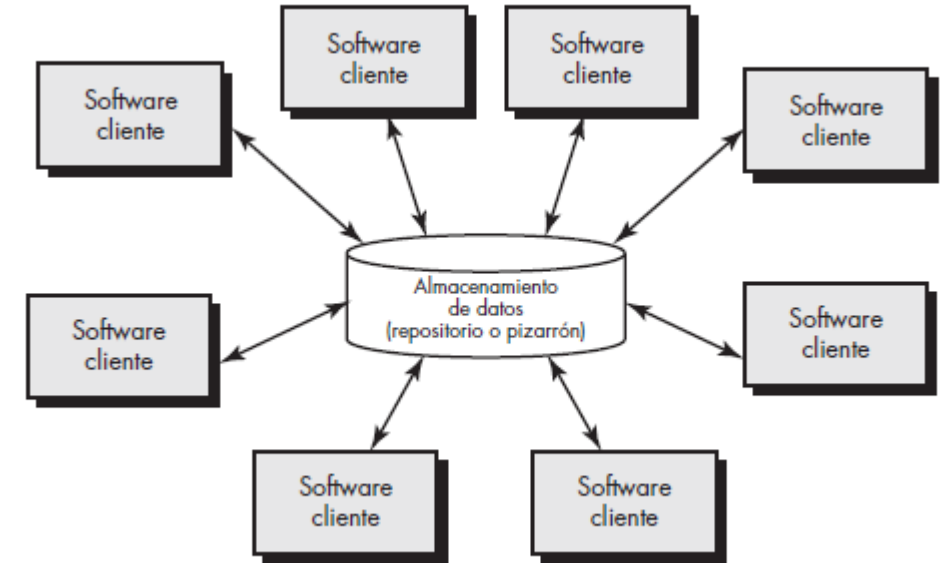
Estilos arquitectónicos y atributos de calidad.

Estilos de arquitectura (centralizado en datos)

La gran mayoría de los sistemas basados en computadoras se clasifica en un numero relativamente pequeño de estilos de arquitectura.

Arquitectura centradas en los datos

- En el centro de esta arquitectura se halla un **almacenamiento de datos** (como un archivo o base de datos) al que acceden con frecuencia otros componentes que actualizan, agrega, eliminan o modifican los datos de cierto modo dentro del almacenamiento.
- El software cliente accede a los datos en forma independiente de cualesquier cambios que estos sufran o de las acciones del software de otro cliente. Una variante de este enfoque transforma **el deposito en un pizarrón que envía notificaciones** al software cliente cuando hay un cambio en datos de interés del cliente.
- Las arquitecturas en datos promueven la **integrabilidad**. Es decir los componentes del software pueden ser cambiados y agregarse otros nuevos, del cliente, a la arquitectura sin problemas con otros clientes (porque los componentes operan en forma independiente). Además pueden pasarse datos entre cliente con el uso de un mecanismo de pizarrón, el cual sirve para coordinar la transferencia de información entre clientes)

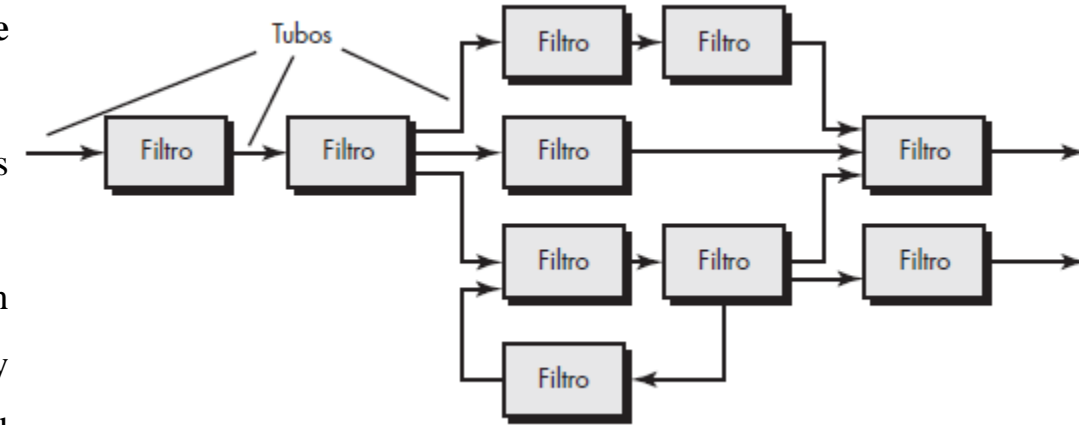


Arquitectura centrada en datos

Estilos de arquitectura (flujo de datos)

Arquitectura de flujo de datos

- Esta arquitectura se aplica cuando **datos de entrada** van a **transformarse** en datos de **salida** a través de una serie de componentes computacionales o manipuladores.
- Un patrón de tubo y filtro tiene un conjunto de **componentes** llamados **filtros**, conectados por tubos que transmiten datos de un componente al siguiente.
- Cada filtro trabaja en forma independiente de aquellos componentes que se localizan arriba o abajo del flujo; se diseña para **esperar una entrada de datos** de cierta forma y **produce datos de salida** (al filtro siguiente) en una forma especificada. Sin embargo, el filtro no requiere ningún conocimiento de los trabajos que realizan los filtros vecinos.
- Si el flujo de datos degenera en una sola línea de transformación, se denomina **lote secuencial**. Esta estructura acepta un lote de datos y luego aplica una serie de componentes secuencias (filtros) para transformarlos.

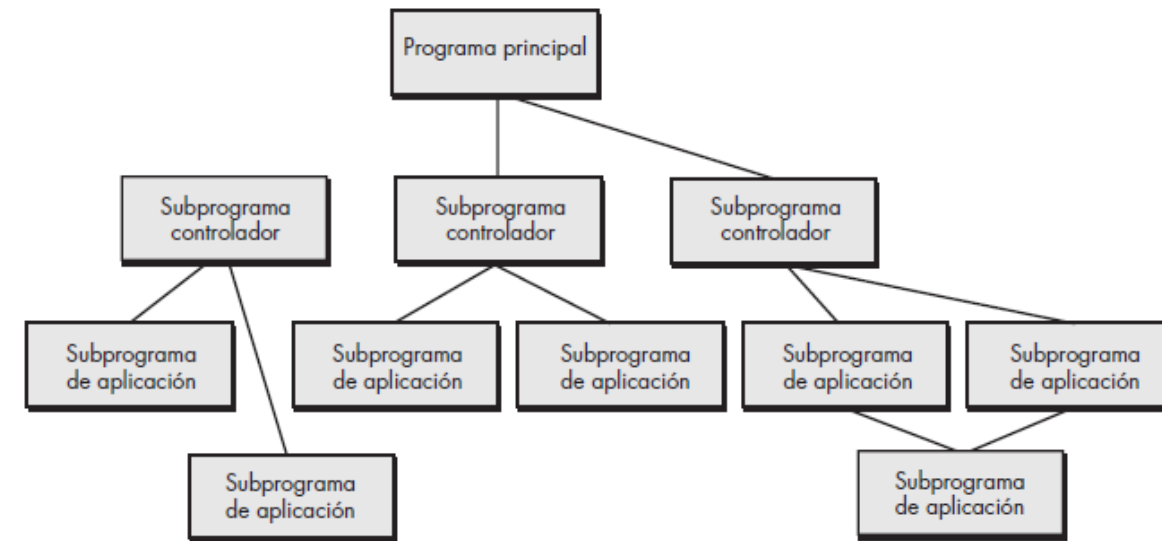


Tubos y filtros

Estilos de arquitectura (llamar y regresar)

Arquitectura de llamar y regresar

- Este estilo arquitectónico permite obtener una estructura de programa que es relativamente fácil de modificar y escalar. Dentro de esta categoría existen varios subestilos:
 - Arquitectura de **programa principal/subprograma**. Esta estructura clásica de programa descompone una función en una jerarquía de control en la que un programa principal invoca cierto número de componentes de programa los que a su vez invocan a otros.
 - Arquitectura de llamada de **procedimiento remoto**. Los componentes de una arquitectura de programa principal/subprograma están distribuidos a través de computadoras en una red.



Arquitectura de programa principal y subprograma

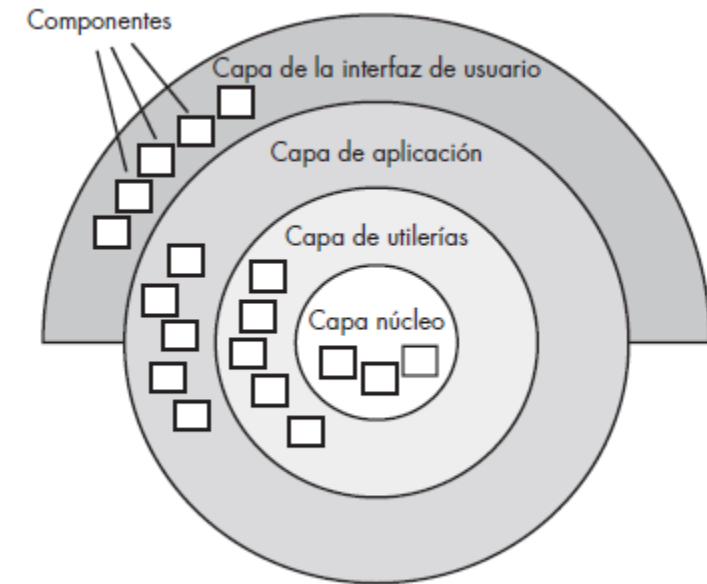
Estilos de arquitectura (orientada a objetos)

Arquitectura orientada a objetos

- Los componentes de un sistema incluyen datos y las operaciones que deben aplicarse para manipularlos. La comunicación y coordinación entre los componentes se consigue mediante la transmisión de mensajes.

Arquitectura en capas

- En la figura se ilustra la estructura básica de una arquitectura en capas. Se define un numero de capas diferentes; cada una ejecuta operaciones que se aproximan progresivamente al conjunto de instrucciones de maquina.
- En la capa externa, los componentes atienden las operaciones de la interfaz de usuario. En la interna, los componentes realizan la interfaz con el sistema operativo. Las capas intermedias proveen servicios de utilerías y funciones de software de aplicación.
- Estos estilos tan solo son un pequeño subconjunto de los que están disponibles. Una vez que la ingeniería de requerimientos revela las características y restricciones del sistema que se va a elaborar, se elige el estilo arquitectónico o la combinación de patrones que se ajuste y es posible diseñar y evaluar estilos arquitectónicos alternativos.



Arquitectura en capas

Patrón arquitectónico

Un patrón es una regla que consta de tres partes, la cual expresa una relación entre un contexto, un problema y una solución. Un patrón sigue el siguiente esquema:

- **Contexto**, es una situación de diseño en la que aparece un problema de diseño.
- **Problema**, es un conjunto de fuerzas que aparecen repetidamente en el contexto,
- **Solución**, Es una configuración que equilibra estas fuerzas. Esta abarca:
 - Estructura con componentes y relaciones.
 - Comportamiento a tiempo de ejecución: aspectos dinámicos de la solución, como la colaboración entre componentes, la comunicación entre ellos, etc.

Partiendo de esta definición, los patrones arquitectónicos se propone como una **descripción de un problema** particular y recurrente de diseño, que aparece en **contextos de diseño específico** y presenta un esquema genérico demostrado con éxito para su solución.

El **esquema de solución** se especifica mediante la descripción de los componentes que la constituyen, sus responsabilidad y desarrollo, así como también la forma como estos colaboran entre sí.

Un patrón arquitectónico de sistema, se puede considerar como una descripción abstracta estilizada de buena practica, que se ensayo y puso a prueba en diferentes sistemas y entornos. De este modo, un patrón arquitectónico debe describir una organización de sistema que ha tenido éxito en sistemas previos.

Un patrón debe describir sobre cuando es y cuando no es adecuado usar dicho patrón, así como sobre las fortalezas y debilidad del patrón.

Diferencias entre estilo y patrones arquitectónico

Un **estilo arquitectónico** es un colección de decisiones de diseño arquitectónico que:

- 1. son aplicables a un **contexto** de desarrollo,
- 2. dado un sistema particular, restringe las decisiones de diseño de arquitectura sobre dicho sistema, y
- 3. garantiza ciertas calidades del sistema resultante

Un **patrón arquitectónico** es una colección de decisiones de diseño arquitectónico que son aplicables a problemas de diseño recurrentes, y que están parametrizados para tener en cuenta los diferentes contextos de desarrollo de software en el que surge el problema.

Con la intención de hacer una comparación mas clara entre estilo arquitectónico y patrón arquitectónico la tabla presenta las diferencias entre estos conceptos, construida a partir del planteamiento de Buschmann et al. (1996)

Estilo Arquitectónico	Patrón Arquitectónico
Sólo describe el esqueleto estructural y general <i>para aplicaciones</i>	Existen en varios rangos de escala, comenzando con patrones que definen la estructura básica de <i>una</i> aplicación
Son independientes del contexto al que puedan ser aplicados	Partiendo de la definición de <i>patrón</i> , requieren de la especificación de un contexto del problema
Cada estilo es independiente de los otros	Depende de patrones más pequeños que contiene, patrones con los que interactúa, o de patrones que lo contengan
Expresan técnicas de diseño desde una perspectiva que es independiente de la situación actual de diseño	Expresa un problema recurrente de diseño muy específico, y presenta una solución para él, desde el punto de vista del contexto en el que se presenta
Son una categorización de sistemas	Son soluciones generales a problemas comunes

Diferencias entre estilo y patrón arquitectónico

Colección de patrones arquitectónicos

La tabla presenta algunos patrones arquitectónicos, además de los atributos que propician y los atributos en conflicto de acuerdo a Buschmann et al (1996)-

Patrón Arquitectónico	Descripción	Atributos asociados	Atributos en conflicto
<i>Layers</i>	Consiste en estructurar aplicaciones que pueden ser descompuestas en grupos de subtarear, las cuales se clasifican de acuerdo a un nivel particular de abstracción.	Reusabilidad Portabilidad Facilidad de Prueba	Desempeño Mantenibilidad
<i>Pipes and Filters</i>	Provee una estructura para los sistemas que procesan un flujo de datos. Cada paso de procesamiento está encapsulado en un componente filtro (<i>filter</i>). El dato pasa a través de conexiones (<i>pipes</i>), entre filtros adyacentes.	Reusabilidad Mantenibilidad	Desempeño
<i>Blackboard</i>	Aplica para problemas cuya solución utiliza estrategias no determinísticas. Varios subsistemas ensamblan su conocimiento para construir una posible solución parcial o aproximada.	Modificabilidad Mantenibilidad Reusabilidad Integridad	Desempeño Facilidad de Prueba
<i>Broker</i>	Puede ser usado para estructurar sistemas de software distribuido con componentes desacoplados que interactúan por invocaciones a servicios remotos. Un componente <i>broker</i> es responsable de coordinar la comunicación, como el reenvío de solicitudes, así como también la transmisión de resultados y excepciones.	Modificabilidad Portabilidad Reusabilidad Escalabilidad Interoperabilidad	Desempeño
<i>Model-View-Controller</i>	Divide una aplicación interactiva en tres componentes. El modelo (<i>model</i>) contiene la información central y los datos. Las vistas (<i>view</i>) despliegan información al usuario. Los controladores (<i>controlers</i>) capturan la entrada del usuario. Las vistas y los controladores constituyen la interfaz del usuario.	Funcionalidad Mantenibilidad	Desempeño Portabilidad

Patrones arquitectónicos y atributos de calidad

Patrón Arquitectónico	Descripción	Atributos asociados	Atributos en conflicto
<i>Presentation-Abstraction-Control</i>	Define una estructura para sistemas de software interactivos de agentes de cooperación organizados de forma jerárquica. Cada agente es responsable de un aspecto específico de la funcionalidad de la aplicación y consiste de tres componentes: presentación, abstracción y control.	Modificabilidad Escalabilidad Integrabilidad	Desempeño Mantenibilidad
<i>Microkernel</i>	Aplica para sistemas de software que deben estar en capacidad de adaptar los requerimientos de cambio del sistema. Separa un núcleo funcional mínimo del resto de la funcionalidad y de partes específicas pertenecientes al cliente.	Portabilidad Escalabilidad Confiabilidad Disponibilidad	Desempeño
<i>Reflection</i>	Provee un mecanismo para sistemas cuya estructura y comportamiento cambia dinámicamente. Soporta la modificación de aspectos fundamentales como estructuras tipo y mecanismos de llamadas a funciones.	Modificabilidad	Desempeño

Diseño arquitectónico (representación del sistema en contexto)

Cuando se comienza el diseño arquitectónico, el software que se va a desarrollar debe situarse en contexto, es decir, definir las entidades externas (otros sistemas, dispositivos, personas, etc) con las que interactúa el software y la naturaleza de dichas interacción. Esta información por lo general se adquiere a partir del modelo de los requerimientos.

Una vez que se ha modelado **el contexto** y descrito todas las interfaces externas del software, se identifica un conjunto de **arquetipos** de arquitectura, sin embargo los arquetipos no dan suficiente detalle para la implementación por lo que se definen y refinan los **componentes** del software que implementan cada arquetipo, siguiendo en forma iterativa hasta que se obtiene una estructura arquitectónica completa.

1. Representación del sistema en contexto

En el nivel de diseño arquitectónico, se usa un diagrama de contexto arquitectónico (DCA) para modelar la manera en que el software interactúa con entidades mas allá de sus fronteras. La estructura general del diagrama de contexto arquitectónico se ilustra en la imagen, en la que se muestran los sistemas que interactúan con el sistema objetivo, el cual esta representado como sigue:

- **Sistemas superiores**, utilizan un objetivo como parte de algún esquema de procesamiento de alto nivel.
- **Sistemas subordinados**, son usados por el sistema objetivo y proveen datos o procesamiento que son necesarios para completar las funciones del sistema objetivo.
- **Sistemas entre iguales**, son los que interactúan sobre una base de igualdad.
- **Actores, entidades** (personas, dispositivos, etc) que interactúan con el sistema objetivo mediante la producción o consumo de información.

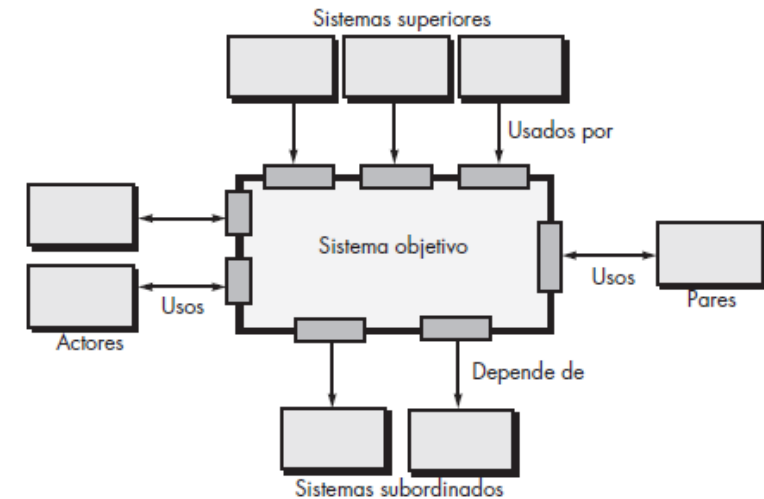


Diagrama de contexto arquitectónico

Diseño arquitectónico (ejemplo de una representación del contexto)

Ejemplo de una representación del sistema en contexto (...continuación)

Cada una las entidades externa que se muestran en una representación del contexto de un sistema se comunica con el sistema objetivo a través de una interfaz (rectángulos pequeños sombreados).

Para ilustrar el empleo del DCA, considere la función de seguridad del hogar del producto “Casa Segura”.

- El controlador general del producto y el sistema basado en internet son superiores respecto de la función de seguridad y se muestran por arriba de la función en la figura. Como parte del diseño arquitectónico, se tiene que especificar los detalles de cada interfaz mostrada tal como se observa en la imagen.
- La función de vigilancia es un sistema entre iguales y en las versiones posteriores del producto usa la función de seguridad del hogar.
- El propietario y los paneles de control son actores que producen y consumen información usada o producida por el software de seguridad.
- Por ultimo los sensores se utilizan por el software de seguridad y se muestran como subordinados de este.

DCA (Diagrama de contexto arquitectónico)

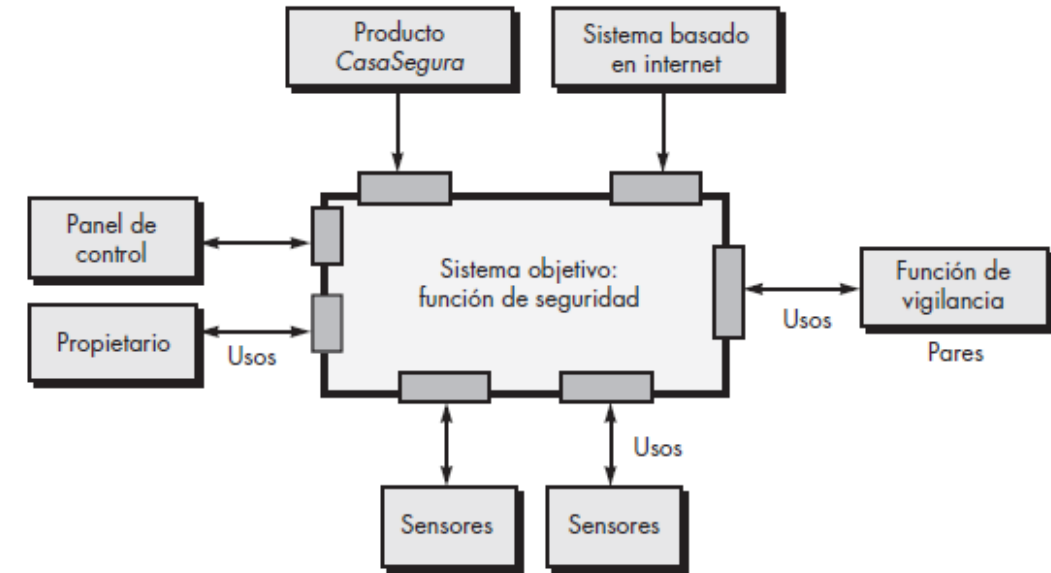


Diagrama de contexto arquitectónico para la función de seguridad de una casa segura

Diseño arquitectónico (definición de arquetipos y ejemplo)

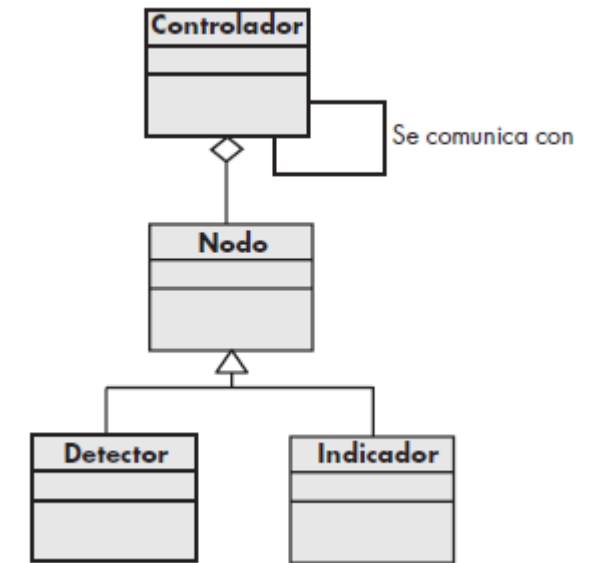
Un **arquetipo** es una **clase o patrón** que representa una abstracción fundamental de importancia crítica para el diseño de una arquitectura para el sistema objetivo. En general, se requiere de un conjunto relativamente pequeño de arquetipos a fin de diseñar sistemas, incluso algo complejos. La arquitectura del sistema objetivo esta compuesta de estos arquetipos, que representan elementos estables de la arquitectura, pero que son implementadas en muchos modos diferentes con base en el comportamiento del sistema

En muchos casos, los arquetipos se obtienen con el estudio de las clases de análisis definidas como parte del modelo de los requerimientos.

2. Representación de los arquetipos a través de un ejemplo

Continuando con el análisis de la función de seguridad de “Casa Segura”, se definen los arquetipos siguientes:

- **Nodo**, representa una colección cohesiva de elementos de entrada y salida de la función de seguridad del hogar. Por ejemplo, un nodo podría comprender: 1) Varios **sensores** (detector) y 2) varios de **indicadores** de alarma (salida).
- **Detector**, abstracción que incluye todos los equipos de detección que alimentan con información al sistema objetivo.
- **Indicador**, abstracción que representa todos los mecanismo (como la sirena de la alarma, luces, campana, etc) que indican que esta ocurriendo una condición de alarma.
- **Controlador**, abstracción que ilustra el mecanismo que permite armar o desarmar un nodo. Si los controladores residen en una red, tienen la capacidad para comunicarse entre si.



Relaciones de UML para los arquetipos de la función de seguridad de “Casa Segura”

Diseño arquitectónico (refinamiento de la arquitectura hacia los componentes)

Conforme se refina la arquitectura hacia los componentes, se comienza a emerger la estructura de sistema. Partiendo del modelo de requerimientos se comienza con las clases descritas, donde estas clases de análisis representan entidades dentro del dominio de aplicación (negocio) que deben enfrentarse dentro de la arquitectura de software. El dominio de aplicación es una fuente para la obtención y refinamiento de los componentes; otra fuente es el dominio de la infraestructura. La arquitectura debe albergar componentes que hagan posible los componentes de la aplicación, pero que no tengan conexión con el dominio de esta, por ejemplo los componentes de administración de memoria, de comunicación, de base de datos y de administración de tareas con frecuencia están integrados en la arquitectura del software.

Las interfaces ilustradas en el diagrama de contexto de la arquitectura implican uno o mas componentes especializados que procesan los datos que fluyen a través de la interfaz. En ciertos casos por ejemplo, una interfaz de usuario grafica debe diseñarse una arquitectura completa con muchos componentes para el subsistema.

3. Representación del refinamiento hacia los componentes

Al seguir con el ejemplo de la función de seguridad de “Casa segura”, debe definirse el conjunto de componentes de alto nivel que se aboque a las funciones siguientes:

- Administración de la comunicación externa

Patrón de diseño

La figura presenta la relación de abstracción existente entre los conceptos de estilo arquitectónico, patrón arquitectónico y patrón de diseño. En ella se presenta el planteamiento de Buschmann et al (1996) que propone el desarrollo de arquitecturas de software como un sistema de patrones y distintos niveles de abstracción.



Relación de abstracción entre estilos y patrones

Los estilos y patrones ayudan al arquitecto a definir la composición y el comportamiento del sistema de software, y una combinación adecuada de ellos permite alcanzar los requerimientos de calidad.

Patrón de diseño

Un patrón de diseño provee un esquema para refinar los **subsistemas o componentes de un sistema de software**, o las relaciones entre ellos.

- Describe la estructura comúnmente recurrente de los componentes en comunicación, que resuelve un problema general de diseño en un contexto particular.
- Son menores en escala que los patrones arquitectónicos, y tienden a ser independiente de los lenguajes y paradigmas de programación. Su aplicación no tiene efectos en la estructura fundamental del sistema, pero si sobre la de un subsistema, debido a que especifica a un mayor nivel de detalle, sin llegar a la implementación, y al comportamiento de los componentes del subsistema.
- La tabla presenta algunos patrones de diseño, junto a los atributos de calidad que propician y los atributos que entran en conflicto con la aplicación del patrón, según Buschmann et al. (1996).

Patrón de Diseño	Descripción	Atributos asociados	Atributos en conflicto
Whole-Part	Ayuda a constituir una colección de objetos que juntos conforman una unidad semántica.	Reusabilidad Modificabilidad	Desempeño
Master-Slave	Un componente maestro (<i>master</i>) distribuye el trabajo a los componentes esclavos (<i>slaves</i>). El componente maestro calcula un resultado final a partir de los resultados arrojados por los componentes esclavos.	Escalabilidad Desempeño	Portabilidad
Proxy	Los clientes asociados a un componente se comunican con un representante de éste, en lugar del componente en sí mismo.	Desempeño Reusabilidad	Desempeño
Command Procesor	Separa las solicitudes de un servicio de su ejecución. Maneja las solicitudes como objetos separados, programa sus ejecuciones y provee servicios adicionales como el almacenamiento de los objetos solicitados, para permitir que el usuario pueda deshacer alguna solicitud.	Funcionalidad Modificabilidad Facilidad de Prueba	Desempeño
View Handler	Ayuda a manejar todas las vistas que provee un sistema de software. Permite a los clientes abrir, manipular y eliminar vistas. También coordina dependencias entre vistas y organiza su actualización.	Escalabilidad Modificabilidad	Desempeño
Forwarder- Receiver	Provee una comunicación transparente entre procesos de un sistema de software con un modelo de interacción punto a punto (<i>peer to peer</i>).	Mantenibilidad Modificabilidad Desempeño	Configurabilidad
Client- Dispatcher- Server	Introduce una capa intermedia entre clientes y servidores, es el componente despachador (<i>dispatcher</i>). Provee una ubicación transparente por medio de un nombre de servicio, y esconde los detalles del establecimiento de una conexión de comunicación entre clientes y servidores.	Configurabilidad Portabilidad Escalabilidad Disponibilidad	Desempeño Modificabilidad
Publisher- Subscriber	Ayuda a mantener sincronizados los componentes en cooperación. Para ello, habilita una vía de propagación de cambios: un editor (<i>publisher</i>) notifica a los suscriptores (<i>suscribers</i>) sobre los cambios en su estado.	Escalabilidad	Desempeño

Diseño arquitectónico

Nivel de Abstracción

Estilo Arquitectónico

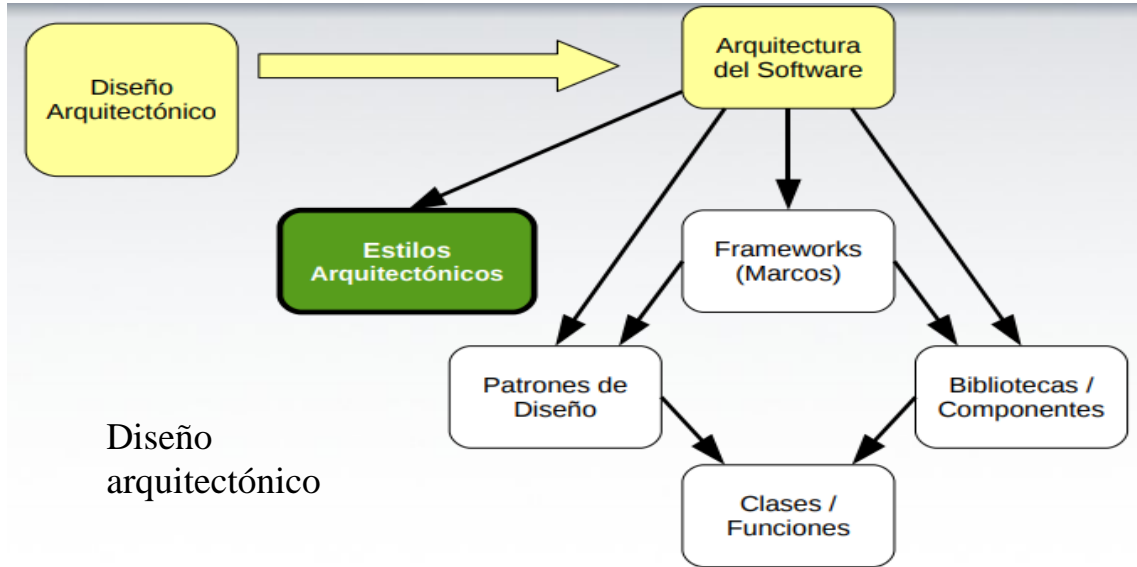
- ✓ Descripción del esqueleto estructural y general para aplicaciones
- ✓ Es independiente de otros estilos
- ✓ Expresa componentes y sus relaciones

Patrón Arquitectónico

- ✓ Define la estructura básica de una aplicación
- ✓ Puede contener o estar contenido en otros patrones
- ✓ Provee un subconjunto de subsistemas predefinidos, incluyendo reglas y pautas para su organización
- ✓ Es una plantilla de construcción

Patrón de Diseño

- ✓ Esquema para refinar subsistemas o componentes



Estilo	Descripción	Atributos asociados	Atributos en conflicto
<i>Datos Centralizados</i>	Sistemas en los cuales cierto número de clientes accede y actualiza datos compartidos de un repositorio de manera frecuente.	Integrabilidad Escalabilidad Modificabilidad	Desempeño
<i>Flujo de Datos</i>	El sistema es visto como una serie de transformaciones sobre piezas sucesivas de datos de entrada. El dato ingresa en el sistema, y fluye entre los componentes, de uno en uno, hasta que se le asigne un destino final (salida o repositorio).	Reusabilidad Modificabilidad Mantenibilidad	Desempeño
<i>Máquinas Virtuales</i>	Simulan alguna funcionalidad que no es nativa al hardware o software sobre el que está implementado.	Portabilidad	Desempeño
<i>Llamada y Retorno</i>	El sistema se constituye de un programa principal que tiene el control del sistema y varios subprogramas que se comunican con éste mediante el uso de llamadas.	Modificabilidad Escalabilidad Desempeño	Mantenibilidad Desempeño
<i>Componentes Independientes</i>	Consiste en un número de procesos u objetos independientes que se comunican a través de mensajes.	Modificabilidad Escalabilidad	Desempeño Integrabilidad

Patrón Arquitectónico	Descripción	Atributos asociados	Atributos en conflicto
<i>Layers</i>	Consiste en estructurar aplicaciones que pueden ser descompuestas en grupos de subtarear, las cuales se clasifican de acuerdo a un nivel particular de abstracción.	Reusabilidad Portabilidad Facilidad de Prueba	Desempeño Mantenibilidad
<i>Pipes and Filters</i>	Provee una estructura para los sistemas que procesan un flujo de datos. Cada paso de procesamiento está encapsulado en un componente filtro (<i>filter</i>). El dato pasa a través de conexiones (<i>pipes</i>), entre filtros adyacentes.	Reusabilidad Mantenibilidad	Desempeño
<i>Blackboard</i>	Aplica para problemas cuya solución utiliza estrategias no determinísticas. Varios subsistemas ensamblan su conocimiento para construir una posible solución parcial ó aproximada.	Modificabilidad Mantenibilidad Reusabilidad Integridad	Desempeño Facilidad de Prueba
<i>Broker</i>	Puede ser usado para estructurar sistemas de software distribuido con componentes desacoplados que interactúan por invocaciones a servicios remotos. Un componente <i>broker</i> es responsable de coordinar la comunicación, como el reenvío de solicitudes, así como también la transmisión de resultados y excepciones.	Modificabilidad Portabilidad Reusabilidad Escalabilidad Interoperabilidad	Desempeño
<i>Model-View-Controller</i>	Divide una aplicación interactiva en tres componentes. El modelo (<i>model</i>) contiene la información central y los datos. Las vistas (<i>view</i>) despliegan información al usuario. Los controladores (<i>controllers</i>) capturan la entrada del usuario. Las vistas y los controladores constituyen la interfaz del usuario.	Funcionalidad Mantenibilidad	Desempeño Portabilidad

Patrones de arquitectura usados en requisitos

ID	Requisitos	Patrón arquitectónico
RF1	Autenticación	Arquitectura Orientada a Servicios
RF2	Autorización	Arquitectura Orientada a Servicios
RF4	Asignación de funcionalidades	MVC
RF5	Envío y recepción de archivos	Arquitectura Orientada a Servicios
RF6	Operaciones CRUD (Create, Read, Update, Delete)	Capas, MVC
RF7	Importación de data desde archivos externos	Tuberías y filtros
RF8	Exportación de datos a archivos externos	Tuberías y filtros
RF9	Envío y recepción de datos de acuerdo a un modelo definido	Arquitectura Orientada a Servicios
RF10	Interconexión entre componentes - servicios	Arquitectura Orientada a Servicios
RF11	Interconexión entre componentes - puertos	Punto a punto
RF12	Abstracción de sistemas de bases de datos relacionales	Capas

Patrones de arquitectura usados en requisitos funcionales

Patrones de arquitectura usados en requisitos no funcionales

ID	Requisitos	Aplicación de patrón arquitectónico	Patrón arquitectónico
RNF1	Disponibilidad	No	-
RNF2	Tiempos de respuesta rápidos (alto rendimiento)	No	-
RNF3	Integridad de la información	No	-
RNF4	Seguridad	No	-
RNF5	Mantenibilidad	Si	Capas, MVC
RNF6	Interoperabilidad	Si	Arquitectura Orientada a Servicios

Bibliografía

Pressman, R. S. (2010). Ingeniería de Software, Un enfoque practico Séptima Edición. Ciudad de México: Mc Graw Hill.

Sommerville. (2011). Ingeniería de Software 9 Edición. Estado de México: Pearson.

UNID Universidad Interamericana para el desarrollo. (2018). Ingeniería de software. Ciudad de México: UNID.