CONCEPTOS DE ARQUITECTURA DE SOFTWARE Y SU RELACION CON LA CALIDAD

[1]. ARQUITECTURA DE SOFTWARE.

Actualmente en la literatura (Bass 1998; Kazman 2001; Hofmeister 2000; Lane 1990; Buschman 1996; Booch 1999; Abowd, 1995), es posible encontrar numerosas definiciones del término **Arquitectura de Software**, cada una con planteamientos diversos. Se hace evidente que su conceptualización sigue todavía en discusión, puesto que no es posible referirse a un diccionario en busca de un significado, y tampoco existe un estándar que pueda ser tomado como marco de referencia.

Sin embargo, al hacer un análisis detallado de cada uno de los conceptos disponibles, resulta interesante la existencia de ideas comunes entre los mismos, sin observarse planteamientos contradictorios, sino más bien complementarios. La intención primordial del análisis no es concluir ni proponer un concepto que englobe todas las ideas planteadas hasta el momento, sino establecer aquellos elementos que no deben perderse de vista al momento de introducirse en el contexto de las arquitecturas de software.

[1.1]. Importancia de la Arquitectura de Software.

La necesidad del manejo de la arquitectura de un sistema de software nace con los sistemas de mediana o gran envergadura, que se proponen como solución para un problema determinado. En la medida que los sistemas de software crecen en complejidad, bien sea por número de requerimientos o por el impacto de los mismos, se hace necesario establecer medios para el manejo de esta complejidad (Hofmeister 1996).

En general, la técnica es descomponer el sistema en piezas que agrupan aspectos específicos del mismo, producto de un proceso de abstracción (Bass 1998) y que al organizarse de cierta manera constituyen la base de la solución de un problema en particular.

De aquí que la mayoría de los autores (Bass 1998; Kazman 2001; Hofmeister 2000; Lane 1990; Buschman 1996; Booch 1999; Abowd, 1995) coinciden en que una arquitectura de software define la estructura del sistema. Esta estructura se constituye de componentes -módulos o piezas de código que nacen de la noción de abstracción, cumpliendo funciones específicas, e interactuando entre sí con un comportamiento definido (Bass 1998; Hayes-Roth 1995; Hofmeister 2000; Buschman 1996; Booch 1999; Abowd 1995).

Los componentes se organizan de acuerdo a ciertos criterios, que representan decisiones de diseño. En este sentido, hay autores que plantean que la arquitectura de software incluye justificaciones referentes a la organización y el tipo de componentes, garantizando que la configuración resultante satisface los requerimientos del sistema (Boehm 1995).

De esta manera, la arquitectura de software puede ser vista como la estructura del sistema en función de la definición de los componentes y sus interacciones (Bass 1998). La práctica ha demostrado que resulta importante extender el concepto considerando los requerimientos y restricciones del sistema (Boehm 1995; Lane 1990), junto a un argumento que justifique que la estructura definida satisface los requerimientos, dándole un sentido más amplio a la definición del término.

La arquitectura de software puede considerarse entonces como el "puente" entre los requerimientos del sistema y la implementación (Hofmeister 2000). Las actividades que culminan en la definición de la arquitectura pueden ubicarse en las fases tempranas del ciclo de desarrollo del sistema: luego del análisis de los requerimientos y el análisis de riesgos, y justo antes del diseño detallado. Desde esta perspectiva, la arquitectura constituye un artefacto de la actividad de diseño (Hofmeister 2000), que servirá de medio de comunicación entre los miembros del equipo de desarrollo, los clientes y usuarios finales, dado que contempla los aspectos que interesan a cada uno (Kazman 2001). Además, pasa a ser la base del diseño del sistema a desarrollar, razón por la cual en la literatura la arquitectura es considerada como plan de diseño del sistema (Hofmeister 2000), debido a que es usada como guía para el resto de las tareas del desarrollo.

De igual manera, serán de particular importancia las propiedades no funcionales del sistema de software, pues influyen notoriamente en la calidad del mismo. Estas propiedades tienen un gran impacto en el desarrollo y mantenimiento del sistema, su operabilidad y el uso que éste haga de los recursos (Buschman 1996). Entre las propiedades no funcionales más importantes se encuentran: modificabilidad, eficiencia, mantenibilidad, interoperabilidad, confiabilidad, reusabilidad y facilidad de ejecución de pruebas (Kazman 2001). Bass (1998) propone que el término "requerimiento no funcional" es disfuncional, debido a que implica que tal requerimiento no existe, o que es una especie de requerimiento que puede ser especificado independientemente del comportamiento del sistema. En este sentido, Bass indica que debe hacerse referencia a atributos de calidad, en lugar de propiedades no funcionales.

Puede observarse que al hablar de arquitectura de software, se hace alusión a:

- La especificación de la estructura del sistema, entendida como la organización de componentes y relaciones entre ellos;
- Los requerimientos que debe satisfacer el sistema y las restricciones a las que está sujeto, así como las propiedades no funcionales del sistema y su impacto sobre la calidad del mismo;
- Las reglas y decisiones de diseño que gobiernan esta estructura y los argumentos que justifican las decisiones tomadas.

Habiendo aclarado el alcance que puede tomar el término arquitectura de software, resulta de gran interés introducir formalmente otros términos que resultan pilares fundamentales dentro del contexto de arquitectura, dado que en torno a ellos gira gran parte del estudio que hasta el momento se ha realizado sobre el tema. Tal es el caso de los componentes, los conectores y las relaciones.

[1.2]. Componentes, conectores y relaciones.

Se entiende por **componentes** los bloques de construcción que conforman las partes de un sistema de software. A nivel de lenguajes de programación, pueden ser representados como módulos, clases, objetos o un conjunto de funciones relacionadas (Buschman 1996). La noción de componente puede llegar a ser muy amplia: el término puede ser utilizado para especificar un conjunto de componentes. Se distinguen tres tipos de componentes (Perry y Wolf, 1992), denominados también elementos, que son:

- Elementos de Datos: contienen la información que será transformada.
- Elementos de Proceso: transforman los elementos de datos.
- Elementos de Conexión: llamados también **conectores**, que bien pueden ser elementos de datos o de proceso, y mantienen unidas las diferentes piezas de la arquitectura.

Una **relación** es la conexión entre los componentes (Buschman 1996). Puede definirse también como una abstracción de la forma en que los componentes interactúan en el sistema a través de

los elementos de conexión. Es importante distinguir que una relación se concreta mediante conectores.

Según Bass (1998), en virtud de que está conformado por componentes y relaciones entre ellos, todo sistema, por muy simple que sea, tiene asociada una arquitectura. Sin embargo, no es necesariamente cierto que esta arquitectura es conocida por todos los involucrados en el desarrollo del mismo. Esto hace evidente la diferencia entre la arquitectura del sistema y su descripción. Esta particularidad propone la importancia de la representación de una arquitectura.

Además de los componentes y conectores, Kazman (2001) contempla las propiedades externamente visibles que comprenden los componentes del software, y las relaciones entre estos. En este sentido, las propiedades externamente visibles hacen referencia a servicios que los componentes proveen, características de desempeño, manejo de fallas, uso de recursos compartidos, etc. En relación a los componentes definidos por la arquitectura de un sistema de software, se tiene la información referente a las interacciones, que son propias de la arquitectura, y que permiten, a nivel de diseño, tomar las decisiones necesarias durante la construcción de un sistema de software.

Kazman (2001) presenta la arquitectura de software como el resultado de decisiones tempranas de diseño, necesarias antes de la construcción del sistema. Según Bass (1998), uno de los aspectos importantes de una arquitectura de software es que, por ser un artefacto de diseño, direcciona atributos de calidad asociados al sistema. Kazman (2001) propone que las arquitecturas facilitan o inhiben estos atributos. Es por ello que se propone el estudio de los atributos de calidad asociados a la arquitectura de un sistema de software, y cuál es su impacto sobre el mismo.

[2]. CALIDAD ARQUITECTÓNICA.

Barbacci (1995) establece que el desarrollo de formas sistemáticas para relacionar atributos de calidad de un sistema a su arquitectura provee una base para la toma de decisiones objetivas sobre acuerdos de diseño y permite a los ingenieros realizar predicciones razonablemente exactas sobre los atributos del sistema que son libres de prejuicios y asunciones no triviales. El objetivo de fondo es lograr la habilidad de evaluar cuantitativamente y llegar a acuerdos entre múltiples atributos de calidad para alcanzar un mejor sistema de forma global.

[2.1]. Atributos de Calidad.

Según Barbacci (1995) la calidad de software se define como el grado en el cual el software posee una combinación deseada de atributos. Tales atributos son requerimientos adicionales del sistema (Kazman 2001), que hacen referencia a características que éste debe satisfacer, diferentes a los requerimientos funcionales. Estas características o atributos se conocen con el nombre de atributos de calidad, los cuales se definen como las propiedades de un servicio que presta el sistema a sus usuarios (Barbacci 1995).

A grandes rasgos, Bass (1998) establece una clasificación de los atributos de calidad en dos categorías:

- Observables vía ejecución: 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 (Availability)	Es la medida de disponibilidad del sistema para el uso (Barbacci 1995).
Confidencialidad (Confidentiality)	Es la ausencia de acceso no autorizado a la información (Barbacci 1995).
Funcionalidad (Functionality)	Habilidad del sistema para realizar el trabajo para el cual fue concebido (Kazman 2001).
Desempeño (Performance)	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). Según Smith (1993), el desempeño de un sistema se refiere a aspectos temporales del comportamiento del mismo. Se refiere a capacidad de respuesta, ya sea el tiempo requerido para responder a aspectos específicos o el número de eventos procesados en un intervalo de tiempo. Según Bass et al. (1998), se refiere además a la cantidad de comunicación e interacción existente entre los componentes del sistema.
Confiabilidad (Reliability)	Es la medida de la habilidad de un sistema a mantenerse operativo a lo largo del tiempo (Barbacci 1995).
Seguridad externa (Safety)	Ausencia de consecuencias catastróficas en el ambiente. Es la medida de ausencia de errores que generan pérdidas de información (Barbacci 1995).
Seguridad interna (Security)	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 2001).

Tabla 1. Descripción de atributos de calidad observables vía ejecución

Es importante destacar que tener conocimiento de los atributos observables, no necesariamente implica que se satisfacen los atributos no observables vía ejecución. Por ejemplo, un sistema que satisface todos los requerimientos observables puede o no ser costoso de desarrollar, así como también puede o no ser imposible de modificar. De igual manera, un sistema altamente modificable puede o no arrojar resultados correctos.

Atributo de Calidad	Descripción
Configurabilidad (Configurability)	Posibilidad que se otorga a un usuario experto a realizar ciertos cambios al sistema (Bosch 1999).
Integrabilidad (Integrability)	Es la medida en que trabajan correctamente componentes del sistema que fueron desarrollados separadamente al ser integrados. (Bass 1998)
Integridad (Integrity)	Ausencia de alteraciones inapropiadas de la información (Barbacci 1995).
Interoperabilidad (Interoperability)	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 1998)
Modificabilidad (Modifiability)	Es la habilidad de realizar cambios futuros al sistema. (Bosch 1999).
Mantenibilidad (Maintainability)	Es la capacidad de someter a un sistema a reparaciones y evolución (Barbacci 1995). Capacidad de modificar el sistema de manera rápida y a bajo costo (Bosch 1999).
Portabilidad (Portability)	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 2001).
Reusabilidad (Reusability)	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 1998).
Escalabilidad (Scalability)	Es el grado con el que se pueden ampliar el diseño arquitectónico, de datos o procedimental (Pressman 2002).
Capacidad de Prueba (Testability)	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 1998).

Tabla 2. Descripción de atributos de calidad no observables vía ejecución.

Bosch (2000) establece que los requerimientos de calidad se ven altamente influenciados por la arquitectura del sistema. Al respecto, Bass (1998) afirma que la calidad del sistema debe ser considerada en todas las fases del diseño, pero los atributos de calidad se manifiestan de maneras distintas a lo largo de estas fases. De esta forma, establecen que la arquitectura determina ciertos atributos de calidad del sistema, pero existen otros atributos que no dependen directamente de la misma. Por ejemplo, la usabilidad de un sistema no está relacionada directamente con la arquitectura del mismo, puesto que los detalles que este atributo envuelve - como el uso de botones o radio-buttons, pantallas intuitivas, etc. - se encuentran casi siempre encapsulados en un componente simple.

Independientemente de esto, es importante tener en cuenta que no puede lograrse la satisfacción de ciertos atributos de calidad de manera aislada. Encontrar un atributo de calidad puede tener efectos positivos o negativos sobre otros atributos que, de alguna manera, también se desean alcanzar (Bass 1998). En su mayoría, los atributos de calidad se pueden organizar y descomponer de maneras diferentes, en lo que se conoce como modelos de calidad. Los modelos de calidad de software facilitan el entendimiento del proceso de la ingeniería de software (Pressman 2002).

Losavio (2003) propone un modelo adaptado de calidad de software para efectos de la evaluación de arquitecturas de software:

La tabla 4 presenta los atributos de calidad planteados por Losavio (2003), que poseen subcaracterísticas asociadas con elementos de tipo arquitectónico.

Característica	Subcaracterística	Elementos de tipo arquitectónico
Funcionalidad	Adecuación	Refinamiento de los diagramas de secuencia
	Exactitud	Identificación de los componentes con funciones responsables de los cálculos
	Interoperabilidad	Identificación de conectores de comunicación con sistemas externos
	Seguridad	Mecanismos o dispositivos que realizan explícitamente la tarea
Confiabilidad	Tolerancia a fallas	Existencia de mecanismos de software para manejar excepciones
	Recuperabilidad	Existencia de mecanismos o dispositivos de software para reestablecer el nivel de desempeño y recuperar datos
Eficiencia	Desempeño	Componentes involucrados en un flujo de ejecución para una funcionalidad
	Utilización de recursos	Relación de los componentes en términos de espacio y tiempo
Mantenibilidad	Acoplamiento	Interacciones entre componentes
	Modularidad	Número de componentes que dependen de un componente
Portabilidad	Adaptabilidad	Presencia de mecanismos de adaptación
	Instalabilidad	Presencia de mecanismos de instalación
	Coexistencia	Presencia de mecanismos que faciliten la coexistencia
	Reemplazabilidad	Lista de componentes reemplazables para cada componente

Tabla 4. Atributos de calidad planteados por Losavio (2003), que poseen subcaracterísticas asociadas con elementos de tipo arquitectónico.

En la literatura es posible encontrar planteamientos en relación al establecimiento de los atributos de calidad y su relación con la arquitectura de software (Bass 2000). Kazman (2001) establece que la arquitectura determina atributos de calidad. Bass (2000) explica la relación existente entre estos, conjuntamente con los problemas y beneficios de la documentación de esta relación.

[2.2]. Relación entre Arquitectura de Software y Atributos de Calidad.

Bass (2000) establece que para alcanzar un atributo específico, es necesario tomar decisiones de diseño arquitectónico que requieren un pequeño conocimiento de funcionalidad. Por ejemplo, el desempeño depende de los procesos del sistema y su ubicación en los procesadores, caminos de comunicación, etc. Por otro lado, establecen que al considerar una decisión de arquitectura de software, el arquitecto se pregunta cuál será el impacto de ésta sobre ciertos atributos; por ejemplo, modificabilidad, desempeño, seguridad, usabilidad, etc.

Por esta razón, Bass (2000) afirma que cada decisión incorporada en una arquitectura de software puede afectar potencialmente los atributos de calidad. Cada decisión tiene su origen en preguntas acerca del impacto sobre estos atributos, y el arquitecto puede argumentar cómo la decisión tomada permite alcanzar algún objetivo. Con frecuencia, el objetivo es un atributo de calidad en particular, por lo que al tomar decisiones de arquitectura de software que afecte a los atributos de calidad, se tienen dos consecuencias:

- Se formula un argumento que explica la razón por la que la decisión tomada permite alcanzar uno o varios atributos de calidad. Esto resulta de gran importancia porque además permite comprender las consecuencias de un cambio de decisión.

- Surgen preguntas sobre el impacto de una decisión sobre otros atributos, que a menudo se responden en el contexto de otras decisiones.

En su planteamiento, Bass (2000) presenta los problemas que existen en relación a la documentación de la relación entre arquitectura de software y atributos de calidad. De igual forma, establecen que la relación entre la arquitectura de software y los atributos de calidad no se encuentra sistemática y completamente documentada, debido a diversas razones:

- Existen atributos de calidad que, luego de ser estudiados durante años, poseen definiciones generalmente aceptadas. Sin embargo, existen algunos que carecen de definición, lo que inhibe el proceso de exploración de la relación en estudio.

- Los atributos no están aislados ni son independientes entre sí. Muchos atributos conforman un subconjunto de otro, es decir, se definen en función de otros atributos que lo contienen. Por ejemplo, el atributo disponibilidad puede ser un atributo por sí mismo; sin embargo, puede ser subconjunto de usabilidad y seguridad.

- El análisis de atributos no se presta a estandarizaciones, puesto que existen diferentes patrones con distintos niveles de profundidad, y resulta complicado establecer cuáles patrones se pueden utilizar para analizar calidad.

- Las técnicas de análisis son específicas para un atributo en particular. Por esta razón es difícil comprender la interacción entre varios análisis de atributos específicos.

A pesar de estas dificultades, Bass (2000) establece que la relación entre arquitectura de software y atributos de calidad tiene diversos beneficios, que justifican su documentación:

- Realza en gran medida el proceso de análisis y diseño arquitectónico, puesto que el arquitecto puede reutilizar análisis existentes y determinar acuerdos explícitamente en lugar de hacerlo sobre la marcha. Los arquitectos experimentados hacen esto intuitivamente, basados en su experiencia en codificación. Por ejemplo, durante el análisis, un arquitecto puede reconocer el impacto de la codificación de una estructura en los atributos de calidad.

- Una vez que el arquitecto entiende el impacto de los componentes arquitectónicos sobre uno o varios atributos de calidad, estaría en capacidad de reemplazar un conjunto de componentes por otro cuando lo considere necesario.

- Una vez que se codifica la relación entre arquitectura y atributos de calidad, es posible construir un protocolo de pruebas que habilitará la certificación por parte de terceros.

Por todo lo expuesto, se reconoce la importancia de la arquitectura de un sistema de software como la base de diseño de un sistema (Kruchten 1999), así como también un artefacto que determina atributos de calidad (Kazman 2001). Ahora bien, es interesante considerar que tanto la arquitectura de un sistema de software como el sistema en sí mismo, se encuentran íntimamente relacionados con su entorno, por lo que es necesario tomarlo en cuenta para efectos del estudio de la calidad y la determinación de los atributos de calidad.

Las características adicionales del sistema o atributos de calidad, están estrechamente relacionadas con el uso que se le dará al sistema propuesto (Kazman 2001), pues es **el contexto** quien define el comportamiento del mismo, sin dejar de lado la funcionalidad (Barbacci 1997). Por ello es necesario analizar el contexto del sistema, dado que de aquí se deriva mucha información relativa a su calidad.

El sistema y su entorno son compañeros en un contrato en el cual cada uno tiene expectativas y obligaciones (Barbacci 1997). En muchos casos, el nivel de compromiso entre el sistema y su entorno no queda establecido de forma clara con el simple análisis de los requerimientos funcionales de un sistema. En este sentido, Barbacci (1997) propone la necesidad del análisis de toda la información disponible, tanto del sistema como del contexto.

Barbacci (1997) plantea un esquema general que permite establecer elementos que deben ser tomados en consideración y que vienen dados por distintos tipos de actividades. Por ejemplo, con base en las actividades del sistema y su importancia, para efectos del sistema y su entorno, es posible determinar que, propiedades como el desempeño o la disponibilidad del sistema, deben ser consideradas como atributos de calidad para [elaborar] la arquitectura del mismo. De igual forma, Barbacci (1997) propone el uso de técnicas como los escenarios, listas de chequeo y cuestionarios, como formas cualitativas de determinar el nivel del contrato, elementos de hardware y la arquitectura de software, teniendo como objetivo principal velar por la calidad del sistema.

A lo largo del proceso de diseño y desarrollo, los atributos de calidad juegan un papel importante, pues con base en estos se generan las decisiones de diseño y argumentos que los justifican (Bass 2000). Dado que la arquitectura de software inhibe o facilita los atributos de calidad (Bass 1998), resulta de particular interés analizar la influencia de ciertos elementos de diseño utilizados para la definición de la misma, determinando sus características. Estos elementos de diseño son los estilos arquitectónicos, los patrones arquitectónicos y los patrones de diseño.

[3]. ESTILOS Y PATRONES.

Bosch (2000) establece que la imposición de ciertos estilos arquitectónicos mejora o disminuye las posibilidades de satisfacción de ciertos atributos de calidad del sistema. Con esto afirman que cada estilo propicia atributos de calidad, y la decisión de implementar alguno de los existentes depende de los requerimientos de calidad del sistema. De manera similar, plantean el uso de los patrones arquitectónicos y los patrones de diseño para mejorar la calidad del sistema. Al respecto, Buschmann (1996) afirma que un criterio importante del éxito de los patrones – tanto arquitectónicos como de diseño - es la forma en que estos alcanzan de manera satisfactoria los objetivos de la ingeniería de software. Los patrones soportan el desarrollo, mantenimiento y evolución de sistemas complejos y de gran escala.

La diferencia entre estilos y patrones arquitectónicos no ha sido aclarada. Bengtsson (1999) plantea la existencia de dos grandes vertientes, que surgen de la discusión de los términos. Shaw y Garlan (1996) utilizan indistintamente los términos estilo arquitectónico y patrón arquitectónico. Por otro lado, Buschmann (1996) establece diferencias sutiles entre ambos conceptos. De cualquier forma, los estilos y los patrones establecen un vocabulario común, y brindan soporte a los ingenieros para conseguir una solución que haya sido aplicada con éxito anteriormente, ante ciertas situaciones de diseño. Además, su aplicación en [la elaboración] de la arquitectura del sistema es determinante para la satisfacción de los requerimientos de calidad.

En vista de la existencia de las dos vertientes, es necesario establecer las posibles diferencias y las razones por las cuales se asume que los términos estilo arquitectónico y patrón arquitectónico son distintos. De igual manera, se busca resaltar los atributos de calidad propiciados tanto por los estilos como por los patrones arquitectónicos y de diseño.

[3.1]. Estilo Arquitectónico.

Shaw y Garlan (1996) definen estilo arquitectónico como una familia de sistemas de software en términos de un patrón de organización estructural, que define un vocabulario de componentes y tipos de conectores y un conjunto de restricciones de cómo pueden ser combinadas. Para muchos estilos puede existir uno o más modelos semánticos que especifiquen cómo determinar las propiedades generales del sistema partiendo de las propiedades de sus partes.

Buschmann (1996) define estilo arquitectónico como una familia de sistemas de software en términos de su organización estructural. Expresa componentes y las relaciones entre estos, con las restricciones de su aplicación y la composición asociada, así como también las reglas para su construcción. Así mismo, se considera como un tipo particular de estructura fundamental para un sistema de software, conjuntamente con un método asociado que especifica cómo construirlo. Éste incluye información acerca de cuándo usar la arquitectura que describe, sus invariantes y especializaciones, así como las consecuencias de su aplicación.

A simple vista, ambas definiciones parecen expresar la misma idea. La diferencia entre los planteamientos de Shaw y Garlan (1996) y Buschmann (1996) viene dada por la amplitud de la noción de componente en cada una de las definiciones. Buschmann (1996) asume como componentes a subsistemas conformados por otros componentes más sencillos, mientras que Shaw y Garlan utilizan la noción de componente como elementos simples, ya sean de dato o de proceso.

En virtud de esto, la diferencia entre ambas definiciones gira en torno al nivel de abstracción, dado que Buschmann (1996) plantea un grado mayor en su concepto de estilo arquitectónico, sugiriendo una estructura genérica para la organización de componentes de ciertas familias de sistemas, independientemente del contexto en que éstas se desarrollen.

La tabla 5 resume los principales estilos arquitectónicos, los atributos de calidad que propician y los atributos que se ven afectados negativamente (atributos en conflicto), de acuerdo con Bass(1998).

Estilo	Descripción	Atributos asociados	Atributos en conflicto
Datos Centralizados	Sistemas en los cuales cierto número de clientes accede y actualiza datos compartidos de un repositorio de manera frecuente.	Integrabilidad Escalabilidad Modificabilidad	Desempeño
Flujo de Datos	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
Máquinas Virtuales	Simulan alguna funcionalidad que no es nativa al hardware o software sobre el que está implementado.	Portabilidad	Desempeño
Llamada y Retorno	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	Mantenibilidad Desempeño
Componentes Independientes	Consiste en un número de procesos u objetos independientes que se comunican a través de mensajes.	Modificabilidad Escalabilidad	Desempeño Integrabilidad

Tabla 5. Estilos Arquitectónicos y Atributos de Calidad.

Un planteamiento reciente, propuesto por Bass (1999), consiste en los estilos arquitectónicos basados en atributos (ABAS), que se establecen como una extensión de la noción de estilo arquitectónico, mediante la asociación de modelos analíticos de atributos de calidad. En este sentido, los autores proponen que estos estilos incluyen un razonamiento cualitativo o cuantitativo, basado en modelos específicos de atributos de calidad.

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 interacción de los tipos de componentes.

Bass (1999) proponen los estilos arquitectónicos como elementos importantes para el diseño, en tanto estos pueden ser elegidos basándose en el entendimiento de las metas de calidad del sistema en construcción. En este sentido, su planteamiento incluye la extensión del concepto de estilo arquitectónico, incluyendo modelos analíticos de atributos de calidad. Un estilo arquitectónico basado en atributos (ABAS) consta de cinco partes (Bass 1999), (ver tabla 6).

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.

Tabla 6. Partes que conforman un estilo arquitectónico basado en atributos (ABAS).

[3.2]. Patrón Arquitectónico.

Buschmann (1996) define patrón como una regla que consta de tres partes, la cual expresa una relación entre un contexto, un problema y una solución. En líneas generales, un patrón tiene 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. Ésta 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, propone los patrones arquitectónicos como 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 responsabilidades y desarrollos, así como también la forma como estos colaboran entre sí.

Así mismo, Buschmann (1996) plantea que los patrones arquitectónicos expresan el esquema de organización estructural fundamental para sistemas de software. Provee un conjunto de subsistemas predefinidos, especifica sus responsabilidades e incluye reglas y pautas para la organización de las relaciones entre ellos. Propone que son plantillas para arquitecturas de software concretas, que especifican las propiedades estructurales de una aplicación - con amplitud de todo el sistema - y tienen un impacto en la arquitectura de subsistemas. La selección de un patrón arquitectónico es, por lo tanto, una decisión fundamental de diseño en el desarrollo de un sistema de software.

Visto de esta manera, el concepto de patrón arquitectónico propuesto por Buschmann (1996) equivale al establecido por Shaw y Garlan (1996) para estilo arquitectónico, quienes tratan indistintamente estos dos términos.

Barbacci (1997) hace la analogía de la construcción de una arquitectura de un sistema complejo como la inclusión de instancias de más de un patrón arquitectónico, compuestos de maneras arbitrarias. La colección de patrones arquitectónicos debe ser estudiada en términos de factores de calidad e intereses, en anticipación a su uso. Esto quiere decir que un patrón puede ser analizado previamente, con la intención de seleccionar el que mejor se adapte a los requerimientos de calidad que debe cumplir el sistema. De manera similar, Barbacci (1997) propone que debe estudiarse la composición de los patrones, dado que ésta puede dificultar aspectos como el análisis, o poner en conflicto otros atributos de calidad. La tabla 7 presenta algunos patrones arquitectónicos, además de los atributos que propician y los atributos en conflicto, de acuerdo a Buschmann (1996).

Patrón Arquitectónico	Descripción	Atributos asociados	Atributos en conflicto
Layers	Consiste en estructurar aplicaciones que pueden ser descompuestas en grupos de subtareas, las cuales se clasifican de acuerdo a un nivel particular de abstracción.	Reusabilidad Portabilidad Facilidad de Prueba	Desempeño Mantenibilidad
Pipes and Filters	Provee una estructura para los sistemas que procesan un flujo de datos. Cada paso de procesamiento está encapsulado en un componente filtro (filter). El dato pasa a través de conexiones (pipes), entre filtros adyacentes.	Reusabilidad Mantenibilidad	Desempeño

Patrón Arquitectónico	Descripción	Atributos asociados	Atributos en conflicto
Blackboard	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
Broker	Puede ser usado para estructurar sistemas de software distribuido con componentes desacoplados que interactúan por invocaciones a servicios remotos. Un componente broker 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
Model-View-Controler	Divide una aplicación interactiva en tres componentes. El modelo (model) contiene la información central y los datos. Las vistas (view) despliegan información al usuario. Los controladores (controlers) capturan la entrada del usuario. Las vistas y los controladores constituyen la interfaz del usuario.	Funcionalidad Mantenibilidad	Desempeño Portabilidad
Presentation-Abstraction-Control	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
Microkernel	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 Confiablidad Disponibilidad	Desempeño
Reflection	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

Tabla 7. Patrones arquitectónicos y atributos de calidad.

Con la intención de hacer una comparación clara entre estilo arquitectónico y patrón arquitectónico, la tabla 8 presenta las diferencias entre estos conceptos, construida a partir del planteamiento de Buschmann (1996).

Estilo Arquitectónico	Patrón Arquitectónico
Sólo describe el esqueleto estructural y general para aplicaciones.	Existen en varios rangos de escala, comenzando con patrones que definen la estructura básica de una aplicación.
Son independientes del contexto al que puedan ser aplicados.	Partiendo de la definición de patrón, 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.

Tabla 8. Diferencias entre estilo arquitectónico y patrón arquitectónico.

[3.3]. 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 (Buschman 1996).

Son menores en escala que los patrones arquitectónicos, y tienden a ser independientes de los lenguajes y paradigmas de programación. Su aplicación no tiene efectos en la estructura fundamental del sistema, pero sí sobre la de un subsistema (Buschman 1996), debido a que especifica a un mayor nivel de detalle, sin llegar a la implementación, el comportamiento de los componentes del subsistema. La tabla 9 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 (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 (master) distribuye el trabajo a los componentes esclavos (slaves). 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.	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 (peer to peer).	Mantenibilidad Modificabilidad Desempeño	Configurabilidad
Client-Dispatcher-Server	Introduce una capa intermedia entre clientes y servidores, es el componente despachador (dispatcher). 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 (publisher) notifica a los suscriptores (suscribers) sobre los cambios en su estado.	Escalabilidad	Desempeño

Tabla 9. Patrones de diseño y atributos de calidad.

La figura 1 presenta la relación de abstracción existente entre los conceptos de estilo arquitectónico, patrón arquitectónico y patrón de diseño. En ella se representa el planteamiento de Buschmann (1996), que propone el desarrollo de arquitecturas de software como un sistema de patrones, y distintos niveles de abstracción.

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.

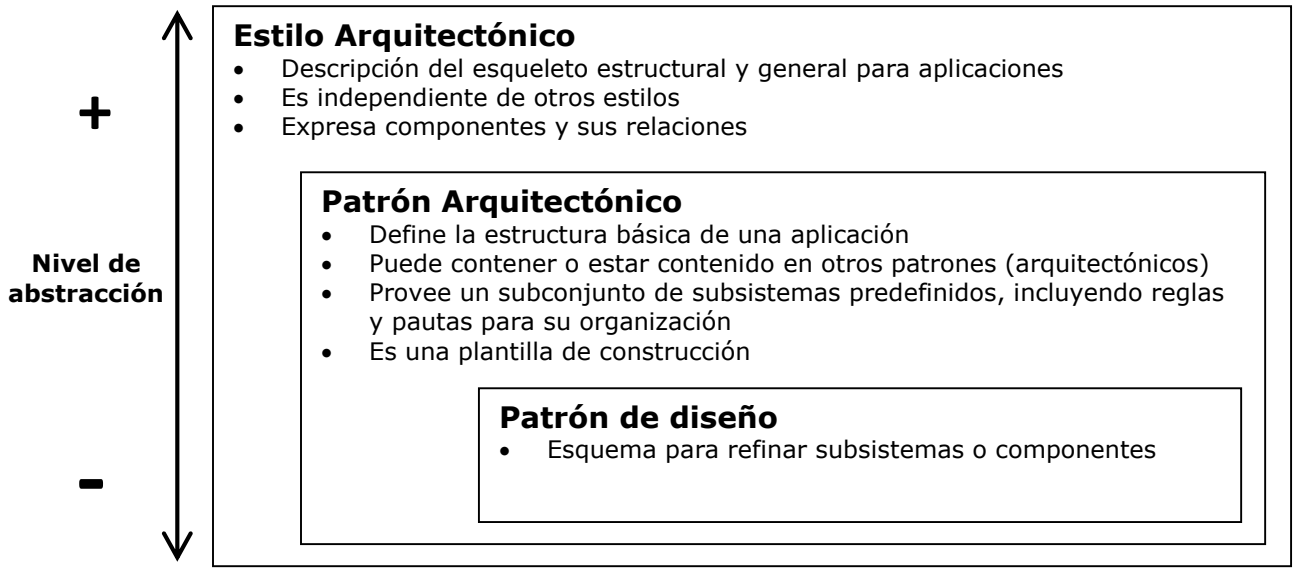

Figura 1. Relación de abstracción entre estilos y patrones.

Ahora bien, la organización del sistema de software debe estar disponible para todos los involucrados en el desarrollo del sistema, ya que establece un mecanismo de comunicación entre los mismos. Tal objetivo se logra mediante la representación de la arquitectura en formas distintas, obteniendo así una descripción de la misma de forma tal que puede ser entendida y analizada por todos los involucrados, con miras a obtener un sistema de calidad. Estas descripciones pueden establecerse a través de las vistas arquitectónicas, las notaciones como UML y los lenguajes de descripción arquitectónica (Bengtsson, 1999).

Las vistas arquitectónicas, a través de distintos niveles de abstracción, resaltan diversos aspectos que conciernen a los involucrados en el desarrollo. Resulta interesante analizar estas perspectivas de una arquitectura, y la forma en que éstas ayudan a todos los involucrados en el proceso de desarrollo a razonar sobre los atributos de calidad del sistema.

---------- FIN DEL DOCUMENTO