

ARQUITECTURAS: ALGUNOS FUNDAMENTOS

Cambios de enfoque

El concepto de Arquitectura, en el ámbito de desarrollo de software, ha ido evolucionando y se ha convertido en un concepto bastante complejo, es decir que abarca una gran diversidad de aspectos del desarrollo. Hace algunos años, estos aspectos se reducían a una misma área, la técnica. De esta forma, si concebimos la multiplicidad de los significados del concepto de arquitectura en términos de puntos de vista, de acuerdo al esquema que vemos en la figura 1:

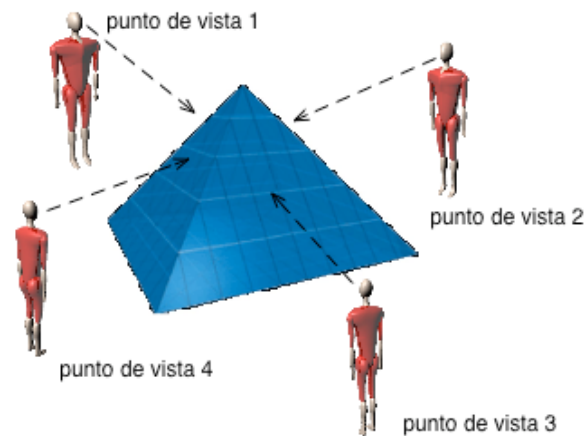


Figura 1 Distintos puntos de vista de una arquitectura

podríamos establecer una equivalencia del tipo:

punto de vista 1 = Separación del sistema en estructuras o componentes

Lo que implica una serie de preguntas del tipo: ¿porqué están separados?, ¿se ejecutan en procesadores distintos?, etc.). De la misma manera tendríamos que si asimilamos punto de vista 2 = ¿cuál es el significado de los enlaces entre los componentes?, derivaríamos en preguntas tales como ¿implican comunicación, control entre ellos, envío de datos?, etc). Punto de vista 3 = ¿cuál es el significado del esquema? nos lleva a ¿porqué uno de los componentes está en un nivel diferente?, ¿eso significa que ese componente llama a los otros?, etc.) y, finalmente, punto de vista 4 = ¿cómo funciona la arquitectura al ejecutarse? Conduce a ¿cómo fluyen los datos y el control a través del sistema?.

Estas son las preguntas que deberíamos responder al pretender describir una arquitectura desde una perspectiva estrictamente técnica, y es evidente que el número de puntos de vista es variable en función de la perspectiva elegida. También remitimos al lector al libro de Bass, Clements y Kazman (*Software Architecture in Practice*, Addison-Wesley, 1998), ya que –aunque algunos autores consideran que los ejemplos que utiliza son demasiado simples o irrelevantes, reconocen sin embargo que la clasificación que ofrece para entender los diversos aspectos de un arquitectura es excelente.

Se añaden nuevos aspectos

A partir de un momento se añaden otros aspectos a los técnicos. En 1995, Philippe Kruchten, de Rational, publicó en la revista *IEEE Software* un artículo bastante conocido (Las 4+1 Vistas del Modelo de Arquitectura) donde comentaba los problemas asociados a querer abarcar con un único diagrama la

esencia de una arquitectura, y añade que cuando nos fijamos en detalle en algún diagrama de ese tipo, resulta evidente que el autor está luchando por incluir mucho más de lo que resultaría razonable. Su modelo de vistas de la arquitectura tenía el siguiente aspecto:

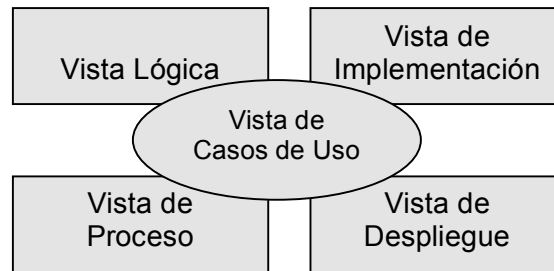


Figura 2 Las 4 + 1 vistas de la arquitectura

Lo que nos resulta llamativo es la forma de referirse a las vistas, diciendo 4 + 1 en vez de 5. El diagrama muestra que la quinta vista, la de los casos de uso, aparece con una representación diferente para indicar que tiene un rol especial en la arquitectura. Contiene algunos escenarios, o casos de uso, para ayudar a descubrir y diseñar la arquitectura en las fases iniciales del proceso al cual adhiere el autor, llamado RUP o Proceso Unificado de Rational. Al representarla diferente –una elipse en vez del rectángulo- nos indica que estos escenarios no eran recogidos previamente en ninguna vista arquitectónica, de ahí la heterogeneidad con la que aparece en la figura. Las 4 vistas restantes capturan los aspectos que comentábamos al comienzo del artículo: la vista lógica contiene elementos tales como paquetes de diseño, subsistemas o clases; la vista de implementación abarca productos tales como el código fuente, los ficheros de datos, componentes y ejecutables; la vista de proceso incluye elementos tales como tareas, hilos de ejecución o procesos y, finalmente, la vista de despliegue muestra cómo los diversos ejecutables y otros componentes de tiempo de ejecución se corresponden con los nodos de proceso en las plataformas subyacentes.

La propuesta de RUP amplía el esquema para abarcar todos los elementos significativos de la arquitectura. Y la pregunta inmediata es, ¿cuáles son esos *elementos significativos*?. A lo que Kruchten (en *The Rational Unified Process: An Introduction*, 1999) responde:

- Las clases principales, es decir aquellas que modelan las entidades principales del dominio considerado
- Los mecanismos arquitectónicos que incorporan comportamiento a esas clases, tales como los de persistencia y comunicación
- Los patrones y los marcos (patterns and frameworks)
- Las capas y subsistemas
- Interfaces
- Los procesos principales, o hilos de control

Y es aquí, en esta enumeración, que vemos incluidos los elementos que darán lugar a una definición más general de la arquitectura: los patrones. Cuando nos hablan de mecanismos –definidos como una clase, o grupo de clases o un patrón que provee una solución común a un problema dado- también se mencionan a los patrones como esquemas que presentan una solución a un problema de diseño recurrente y que se presenta en situaciones específicas del diseño.

Patrones arquitectónicos

La diferencia principal entre mecanismo y patrón es que estos últimos son esquemas más generales que los mecanismos, es decir que describen interacciones amplias de elementos abstractos de diseño que permiten al arquitecto o diseñador pensar en un problema complejo mediante una abreviatura intuitiva.

Son los patrones los que permitirán hablar posteriormente de la arquitectura en términos de abarcar también las *decisiones más importantes* en relación con el sistema a desarrollar. Y a primera vista resulta chocante ver colocados en un mismo conjunto a componentes, subsistemas, ejecutables y procesadores con *decisiones*.

Se empieza a decir, entonces, que la caja de herramientas para resolver problemas arquitectónicos está progresivamente más llena de patrones y mecanismos. Para ir llenando la caja de herramientas aparecen libros como los de Buschmann y otros (*Pattern-Oriented Software Architecture: A System of Patterns*, 1996) que con el tiempo quedó como primer volumen de una saga que añadió el de Schmidt y otros en el 2000, con el mismo título y subtítulo *Patterns for Concurrent and Networked Objects*.

La extensa lista de patrones arquitectónicos abarca –por ejemplo– el clásico patrón arquitectónico de Capas, que ayuda a estructurar una aplicación, descomponiéndola en grupos de subtareas en los que cada grupo es un nivel particular de abstracción. El componente externo al patrón –llamado cliente– usa la capa más alta, es decir de mayor nivel de abstracción, y es ésta la que se ocupa de llamar a la siguiente, que a su vez puede requerir llamar a otra y así sucesivamente a lo largo de todas las capas.

Otro clásico patrón arquitectónico es el de Modelo-Vista-Controlador. Esta solución, que fué originariamente adoptada en Smalltalk 80, es otro buen ejemplo de patrón arquitectónico, ya que propone dividir una aplicación interactiva en tres componentes. El *Modelo* contiene la funcionalidad principal y los datos. Las *Vistas* muestra la información al usuario, y los *Controladores* se ocupan de la entrada del usuario. La interfaz de usuario se compone de las vistas y controladores. Hay además un mecanismo para asegurar la consistencia entre la interfaz de usuario y el modelo. Como puede verse, los patrones arquitectónicos ofrecen un conjunto de subsistemas predefinidos, en los que se especifica sus responsabilidades y se incluyen reglas y orientaciones para organizar las relaciones entre ellos. Los patrones son soluciones para elaborar piezas de la arquitectura que permiten ir armando progresivamente el rompecabezas.

La arquitectura ejecutable

Durante algún tiempo, la gente de Rational insistían en el hecho de que la Arquitectura era algo más que un plano (es decir, algo más que el conjunto de las 4 + 1 vistas que proponen para modelar el sistema). Para efectuar esta afirmación, añadían que era necesario validar la arquitectura para evaluar su calidad en términos de viabilidad, rendimiento, flexibilidad y robustez y, por lo tanto había que construirla. Entonces concluían que había que modelarla, construirla, validarla y luego tomarla como base del desarrollo. Es decir que ese prototipo arquitectónico debía implementar las decisiones más importantes de diseño a efectos de poder validarlas. Y el concepto de Arquitectura abarcaba tanto el plano como el prototipo, con lo cual empezaba a resultar un tanto confuso, ya que no se trataba solamente del modelo sino también de la implementación y el concepto empezaba a desdibujarse.

En las últimas versiones de su producto RUP, añaden un concepto adicional que clarifica la cosa. Ahora hablan de *Arquitectura Ejecutable*, que es según su glosario “una implementación parcial del sistema, construido para demostrar algunas funciones y propiedades seleccionadas del sistema, en particular aquellas que satisfacen requisitos no funcionales. Se construye durante la fase de elaboración¹ para mitigar los riesgos relacionados con el rendimiento, funcionamiento, capacidad, fiabilidad y otros aspectos, de tal manera que la capacidad funcional completa del sistema puede llevarse a la fase de construcción sobre una base sólida, sin temor de desajustes”.

Los estándares

Una definición formal, incorporada en el estándar de ANSI/IEEE 1471-2000 nos dice que la arquitectura es “la organización fundamental de un sistema, incorporado en sus componentes, las relaciones entre ellos y

¹ La de elaboración en una de las 4 fases en que RUP divide el desarrollo: Inicio, Elaboración, Construcción y Transición.

con el entorno y los principios que rigen su diseño y evolución”, con lo cual se añaden –a la estructura– también los principios. Retoma esta definición The Open Group, una organización que define un marco arquitectónico, disponible gratuitamente para cualquier empresa que quiera utilizarlo para uso interno. Este marco (TOGAF: <http://www.opengroup.org/architecture/>) abarca lo que denominan la Arquitectura de la Empresa y que incluye:

- Una arquitectura de negocio (o de procesos de negocio): define la estrategia de negocio, su forma de dirección, la organización y los procesos clave de negocio
- Una arquitectura de aplicaciones: esta clase de arquitectura suministra un plano de los sistemas de aplicación individuales que serán desplegados, sus interacciones y sus relaciones con los procesos de negocio principales de la organización
- Una arquitectura de datos: describe la estructura de los bienes lógicos y físicos de los datos de la organización y de los recursos de gestión de datos
- Una arquitectura tecnológica: describe la infraestructura de software dirigida a soportar el despliegue de las principales aplicaciones críticas de la organización. Este tipo de software se denomina a veces ‘middleware’.

La arquitectura de aplicaciones y la de datos conforman lo que se llama la Arquitectura de Sistemas de Información, y coincide con lo que hemos estado definiendo hasta ahora como arquitectura.

El Grupo de Trabajo en Arquitecturas de IEEE (IEP 1471) da una proyección bastante general al concepto, ya que una arquitectura es el concepto de más alto nivel de un sistema y su entorno. También abarca el encaje con la integridad del sistema, con las restricciones económicas, con las preocupaciones estéticas y con el estilo. Es decir que –al revés que en sus comienzos– el enfoque deja de ser ‘hacia adentro’ y comienza a tener en cuenta al sistema como un todo, con el entorno de usuario y de desarrollo, es decir un enfoque ‘hacia afuera’.

Más allá de la arquitectura

Esta expansión hacia afuera continúa progresando, y en algunos de los últimos trabajos (Hohmann, *Beyond Software Architecture: Creating and Sustaining Winning Solutions*, 2003) se reconoce que, a diferencia de las definiciones previas, que se centran en los aspectos técnicos de la arquitectura, se debe poner el acento en los aspectos humanos y de negocio, que también forman parte de la gran imagen arquitectónica. De esta manera, podríamos asociar en la figura 1 el punto de vista 1 = enfoque técnico, punto de vista 2 = enfoque humano y punto de vista 3 = enfoque de negocio o de marketing para tener una visión mucho más amplia de la arquitectura.

Es cierto, tal como afirma el autor, que aunque pongamos el énfasis en la creación inicial y las primeras versiones de la arquitectura, invertiremos la mayor parte del tiempo en trabajar –reelaborar– la arquitectura existente. La evolución de la arquitectura puede resultar mucho más fascinante e interesante que la creación de su versión final. Es sólo a través de la evolución que podemos constatar dónde hemos tenido éxito o no, sobre todo cuando la evolución se basa en la realimentación directa por parte de los clientes.

La actual tendencia –tanto de marketing como en el caso de las metodologías ágiles, en especial FDD– es a hablar de *características* (features) de un producto software, como algo que el producto hace o debería hacer (es decir, las antiguas funcionalidades). Pero esta nueva denominación no es simplemente un cambio de palabra –que puede gustar mucho a los informáticos– sino que hace que se pueda hablar de un producto desde el punto de vista de los usuarios o de marketing –externo– y no de aspectos internos que entiende principalmente el desarrollador, como es el caso de las funcionalidades.

Por eso ahora puede hablarse de la *capacidad* de una arquitectura para referirse a la habilidad de la misma para soportar un conjunto determinado de características. La capacidad de una arquitectura resulta evidente cuando le decimos a marketing que una colección relacionada de características –o un conjunto de ellas que no parecen relacionadas a primera vista, pero que luego sí lo están por detalles de implementación– es difícil o imposible de implementar dentro de nuestra actual arquitectura. Esto implica una interacción continua entre la madurez y evolución de las arquitecturas.