UCAB UNIVERSIDAD CATÓLICA ANDRÉS BELLO

ARQUITECTURA DE *SOFTWARE*

Tabla de contenido

Objetivo

Valorar la arquitectura de *software* como uno de los productos del diseño del *software* más importante.

Introducción

La complejidad de los sistemas de *software* de hoy día nos obliga a subir el nivel de abstracción cuando pensamos en su diseño para cumplir con las cualidades arquitectónicas solicitadas. Y es aquí cuando surge el concepto de "arquitectura de *software*", el cual se maneja de manera similar al concepto de arquitectura en la construcción civil.

Si observamos las obras civiles que nos rodean, por ejemplo, la UCAB, podremos comprobar que se caracteriza por espacios amplios que facilitan la circulación de volúmenes importantes de personas; casi siempre de obra limpia para facilitar su mantenimiento y aguantar "el maltrato" de sus usuarios; amplios accesos, etc. El diseño es tal que satisface la funcionalidad deseada y además garantiza su perpetuidad en el tiempo.

Pues bien, en el mundo del *software* (en sistemas complejos) se manejan los mismos principios. Solo que aquí se habla de confiabilidad, eficiencia, seguridad, etc. Organizaciones tales como los bancos, industrias de mediano y gran tamaño, cuentan con plataformas de Tecnologías de la Información (TI) muy complejas, siendo clave que estas funcionen de manera óptima para su supervivencia. Por ello, es fundamental tener un dominio sobre las arquitecturas de *software* de sus diferentes sistemas; es decir, contar con su representación y hacerle seguimiento a su evolución. Aquí vamos a reflexionar sobre su valor.

Arquitectura de *software*

La arquitectura emerge como una parte crucial del proceso de diseño. La arquitectura del *software* se contempla solo en sistemas complejos. La vista arquitectónica de un sistema es abstracta, no considera detalles de implementación, algoritmos o estructura de datos, se concentra en el comportamiento e interacción entre elementos tipo "caja negra".

¿Por qué hablamos de vistas arquitectónicas? ¡Porque la arquitectura es multidimensional!

Ya se explicó, en el tema anterior, que se necesita más de un diagrama para representarla. Los sistemas de *software* se construyen para satisfacer las necesidades del negocio. La arquitectura es el conjunto de estructuras necesarias para determinar el sistema: contempla elementos de *software*, relaciones entre ellos y propiedades de ambos. Las decisiones sobre la arquitectura pueden ser o no tempranas, pero esta implica una toma de decisiones. Si estas decisiones las tomamos sobre un diagrama serán menos costosas que si las tomamos sobre un código ya elaborado.

1. **Ciclo de negocio de la arquitectura:**
    - La arquitectura de *software* es un resultado de las influencias técnicas, de negocio y sociales.
    - A su vez, su existencia afecta a los ambientes técnicos, de negocio y sociales, que subsecuentemente afectan a la futura arquitectura. Es un ciclo.

Vamos a analizarlo por partes:

Comenzamos revisando porque es "resultado de las influencias técnicas, de negocio y sociales":

Técnicas:

- Por ejemplo, en Venezuela contamos con bancos que tienen una arquitectura basada en mainframe. Esta fue una decisión técnica muy acertada en su momento, sobre todo en los bancos más antiguos. Decisión

tomada por banqueros de la vieja usanza. Esto nos lleva a una arquitectura de *software* centralizada.

- Pero también contamos con bancos más jóvenes, que basan su plataforma en estaciones de trabajo muy poderosas. También una decisión muy acertada, tomada por banqueros muy jóvenes (en su momento) audaces, etc. Esta nos lleva a una arquitectura de *software* distribuida.

Del negocio:

- Los bancos deben decidirse por arquitecturas robustas y muy seguras que les garanticen disponibilidad siempre y eviten el fraude.
- En cambio, las universidades hasta ahora solo necesitaban arquitecturas que les garantizaran un trabajo de oficina (control de estudios, inscripciones, nómina, contabilidad etc.) y que soportaran sus procesos administrativos.

Sociales:

- Nosotros contamos con universidades que surgieron del mundo empresarial. Por ello, tienen una plataforma muy robusta, donde la parte financiera es muy importante para su existencia.
- Tenemos otras, como la UCAB, donde lo financiero no es lo más relevante. La UCAB es la universidad que más becados tiene.
- En las públicas la parte financiera no tiene gran complejidad, salvo el control de gastos, más no los ingresos, porque son financiadas por el Estado. Estos tres modelos de negocio exigen arquitecturas de *software* diferentes.

¡Ahora vamos a analizarlo al revés!

"Porque su existencia afecta a los ambientes técnicos, de negocio y sociales que subsecuentemente afectan a la futura arquitectura":

Técnicos:

- En los bancos que tienen una arquitectura basada en *mainframe* todos los procesos de negocio son lentos. Es decir, si vas a pagar algo con la página

web de estos bancos tienes que introducir: usuario, clave, afiliarte, por coordenadas, etc. ¡Un trasatlántico, lento pero seguro!

- ¡En los bancos con plataforma distribuida todo es más fácil, más ágil! Pero ojo, esto tiene sus problemas: más vulnerabilidad, más fraudes, etc. Un venadito, muy rápido, pero…
- Si la arquitectura es muy segura, como la basada en *mainframe*, la institución ofrece servicios seguros pero lentos.
- Si la arquitectura es descentralizada, distribuida, con estaciones de trabajo por ejemplo, sus procesos son más ágiles y ofrecen respuestas al cliente más rápidamente; sin embargo, hay menos control, más inmadurez, menos seguridad, etc.

Del negocio y sociales:

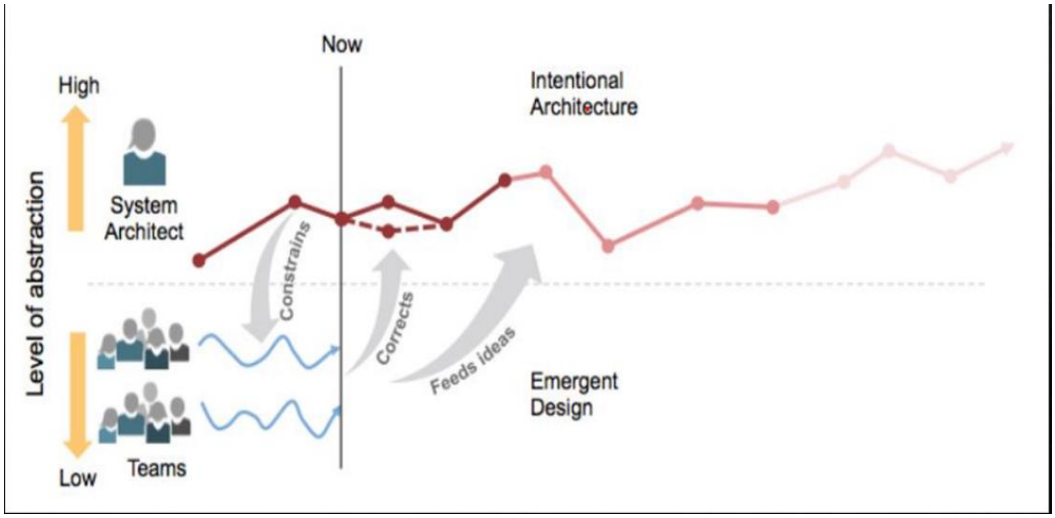
- Las universidades con plataformas poderosas pueden tener una cobertura más amplia con su arquitectura y ofrecer procesos académicos novedosos.
- Las que no lo tienen ofrecen una gran ayuda social, pero sus estudiantes tienen que hacer muchas colas.

Muchas personas están involucradas en la elaboración de la arquitectura del *software*. Ellas son los *stakeholders*. Estos tienen diferentes intereses que desean maximizar con el sistema, cosas como cierto comportamiento en tiempo de ejecución, desempeño óptimo de alguna pieza de *hardware*, fácil uso para el cliente *time to market*, etc. El problema de fondo es que cada *stakeholder* tiene un interés propio y estos pueden entrar en contradicción.

Además de las metas organizacionales, la arquitectura también se ve influenciada por la naturaleza de la organización desarrolladora. Digamos la gerencia de sistemas/TI de cada organización.

Las decisiones arquitectónicas también se ven afectadas por la educación y tradición del arquitecto, quien se expone solo a usar los mecanismos o decisiones arquitectónicas que le han sido exitosas. Este *background* del arquitecto se conoce como entorno técnico.

Según Richards y Ford (2020), con el desarrollo ágil la arquitectura está en continua evolución (figura 1):



Fuente: desarrollo ágil de la arquitectura. Extraída de Richards y Ford (2020)

El proceso de desarrollo del *software* es el nombre que se le da a la organización y gerencia de las actividades que implican desarrollar *software*. Una de sus actividades es el diseño. En cualquier subproceso de diseño habrá múltiples diseños candidatos a ser considerados. Evaluar la arquitectura sobre las cualidades que soporta es esencial para asegurar que el sistema satisfaga los intereses de los *stakeholders*.

2. Estructura:

Asumamos que la arquitectura es un conjunto de componentes y las conexiones entre ellos; o que la arquitectura de un *software* o un programa es la estructura o estructuras del sistema, la cual comprende los elementos de *software*, las propiedades visibles de estos elementos y las relaciones entre ellos. Por ejemplo, clases y pases de mensajes o páginas y enlaces.

Estas definiciones implican que:

- La arquitectura contempla información sobre estos elementos de *software* y cómo se relacionan.
- Ella es la abstracción del sistema. Los detalles de implementación no son arquitectónicos.

- La arquitectura cubre más de una estructura. Una sola no cubre todos los aspectos arquitectónicos.
- El comportamiento de cada elemento es parte de la arquitectura, y cómo estos interactúan entre sí.

Además:

- Es un diseño de alto nivel.
- Las diferentes estructuras proveen información sobre puntos críticos para la satisfacción de las cualidades arquitectónicas. Por ello, este conjunto de estructuras da orientaciones sobre su comportamiento y evolución en el tiempo.

Entre sus estructuras tenemos:

- **Las estáticas/modulares**: la estructura que se refiere a sus unidades de implementación, módulos, tales como los Diagramas de componentes o de clases.
- **Las dinámicas**: se enfocan en representar el comportamiento de la arquitectura en tiempo de ejecución, tales como los Diagramas de secuencia.
- **Las de asignación:** se refieren al *mapping* entre las estructuras de *software* y los ambientes organizacionales de desarrollo, de instalación o ejecución. Tales como los Diagramas de despliegue.

3. Importancia de la arquitectura:

Es importante porque:

- Facilita la comunicación entre *stakeholders*.
- En una decisión temprana de diseño.
- Es una abstracción transferible del sistema. ¿Cuántas veces no reutilizamos un diseño arquitectónico?
- Orienta el desarrollo: sus diagramas nos sugieren por dónde empezar.
- Restringe el vocabulario de las alternativas de diseño.
- Habilita o inhibe atributos de calidad. ¡Lo más importante! ¡Un diseño inseguro genera un *software* inseguro!

Por ejemplo:

- Si queremos un buen desempeño le prestamos atención a los componentes, a los recursos compartidos, al volumen de comunicaciones entre elementos.
- Si queremos modificabilidad le prestamos atención a los elementos que mayores cambios tendrán en el tiempo.
- Si queremos seguridad debemos prestar atención a la comunicación entre los elementos.
- Si queremos escalabilidad debemos prestar atención en colocar unidades de alto reemplazo, a fin de disminuir el recódigo lo más posible.
- Si queremos reutilización debemos disminuir el acoplamiento.

Atención: la arquitectura no lo resuelve todo. Una pobre implementación afecta también significativamente. La calidad es algo para mantener a lo largo del ciclo de vida. Una buena arquitectura es necesaria, más no suficiente.

4. Características de una buena arquitectura:

Recordemos: la arquitectura muestra la correspondencia entre los requerimientos de sistemas y los elementos del sistema construido. Por ello debe cumplir con:

- **Conformidad funcional:** según Pressman y Maxim (2020), una arquitectura de calidad debe implementar todos los requisitos explícitos contenidos en el modelo de análisis y debe acomodar todos los requisitos implícitos que desea el cliente.
- **Adaptabilidad:** esta característica la propone Sommerville (2020), al señalar que ella mide cuán fácil es hacer cambios en una arquitectura; de seguro esto implica componentes poco acoplados.
- **Modularidad:** sin considerar el enfoque de diseño utilizado (estilo arquitectural) (Pfleeger, 1998), el proceso de descomposición separa el diseño en las partes que lo componen, llamadas módulos o componentes.
- **Entendible:** Sommerville (2020) señala que antes de hacerle un cambio a un sistema, este debe ser entendido.

- **Cohesión:** es una consecuencia del ocultamiento de la información.

    - Un módulo con cohesión (Pressman, 2020) realiza solamente una tarea, requiriendo poca interacción con el resto de los procedimientos que se realizan en el resto del sistema de *software*.
    - Según Sommerville (2020), la cohesión es deseable debido a que una unidad (componente) representa una parte simple de solución. Si es necesario cambiar el sistema, la parte correspondiente está en un solo lugar y lo que se desee hacer con él estará encapsulado en él.
    - Según Pfleeger (1998), la meta es hacer que los componentes sean lo más cohesivos posible.

Los arquitectos pueden colaborar en la identificación de los requisitos, pero una de sus responsabilidades clave es definir, descubrir y analizar todas las cosas que debe hacer el *software* que no están directamente relacionadas con la funcionalidad: las características arquitectónicas. Pueden ser llamadas como no funcionales, de calidad o suplementarias. Preferimos llamarlas "características de la arquitectura", porque describen preocupaciones críticas para el éxito de la arquitectura y, por lo tanto, del sistema en su conjunto, sin descontar su importancia.

Una característica de la arquitectura es que cumple con tres criterios:

a) **Especifica una consideración de diseño que no es de dominio**: por ejemplo, la eficiencia muchas veces no está explícita en ninguna especificación de requerimiento, pero todo el mundo espera una aplicación rápida y que consuma pocos recursos.
b) **Influye en algún aspecto estructural del diseño**. Por ejemplo, la seguridad puede requerir algoritmos cifrados, claves, logueo o toda una capa de seguridad.
c) Es **fundamental o importante** para el éxito de la aplicación.

Estas características arquitectónicas a menudo interactúan entre sí, lo que lleva al uso generalizado del término *trade-off* entre los arquitectos. Las características arquitectónicas pueden ser de tres tipos (Richards y Ford, 2020):

- **Operacionales** (en mi opinión son calidad externa, según la definición de ISO25010): las características operacionales cubren capacidades como rendimiento, escalabilidad, elasticidad, disponibilidad y confiabilidad.
- **Estructurales** (para mi es calidad interna, según la definición de ISO25010): los arquitectos deben preocuparse por la estructura del código. En muchos casos, el arquitecto tiene la responsabilidad exclusiva o compartida de los problemas de calidad del código, como un buen modularidad, acoplamiento controlado entre componentes, código legible y una serie de otras evaluaciones de calidad interna.
- **Transversales** (ni lo uno ni lo otro): ¿de qué estamos hablando? De características como: accesibilidad, persistencia, autenticación, autorización, legal, intimidad, seguridad, soporte, usabilidad.

El mayor problema radica en el hecho de que cada característica de la arquitectura a menudo tiene un impacto en otras. Por ejemplo, si un arquitecto desea mejorar la seguridad, es casi seguro que tendrá un impacto negativo en el rendimiento: la aplicación debe realizar más encriptación sobre la marcha, direccionamiento indirecto para ocultar secretos y otras actividades que potencialmente degradan el rendimiento. Así, los arquitectos rara vez se encuentran en una situación en la que pueden diseñar un sistema y maximizar cada característica de la arquitectura sin afectar a otra. Las decisiones se reducen a *trade-off* entre varias características.

Cierre

Uno de los productos más importantes del diseño es la representación de la arquitectura de *software*. Ella nos da una visión holística de nuestro(s) sistema(s) de *software*. Es consecuencia de decisiones técnicas, de negocio y sociales. Al mismo tiempo, una vez que esta arquitectura esté en operación, afectará nuestras próximas decisiones. De allí su valor.

El arquitecto debe garantizar que se dé un mejor *tradde-off* en relación a las cualidades arquitectónicas deseadas; además debe hacerle seguimiento a su evolución, tratando en lo posible que sea un diseño adaptable, modular, entendible y cohesivo. ¡Nada fácil!

Referencias

Pfleeger, S. (1998). *Software Engineering: Theory and Practice* (1st Edition). Prentice Hall.

Pressman, R. y Maxim, B. (2020). *Software Engineering: A Practitioner's Approach.* McGraw Hill.

Richards, M. & Ford, N. (2020). *Fundamentals of Software Architecture*: *An Engineering Approach.* O'Reilly Inc.

Sommerville, I. (2020). *Engineering Software Products: An Introduction to Modern Software Engineering*. Pearson.

Referencias de las imágenes

Richards, M. & Ford, N. (2020*). Desarrollo ágil* [Imagen]. Disponible en: Fundamentals of Software Architecture: An Engineering Approach. O'Reilly Inc.