Modernización de Arquitecturas del Software

Basado en el podcast ***SE Radio 638: Nick Tune and Jean-Georges Perrin on Architecture Modernization***

David Covián Gómez – UO295168

Daniel Álvarez Blanco – UO290321

Germán García de la Llana – UO283016

Arquitectura del Software

Modernización de la arquitectura del software

La modernización de la arquitectura del software es el proceso de actualizar sistemas software antiguos para adaptarlos a las necesidades actuales del negocio y la tecnología. La modernización consiste en eliminar esas desventajas de los sistemas antiguos usando prácticas modernas, tanto en lo que hacemos como en cómo pensamos. Este proceso no repercute únicamente sobre el código, sino sobre la organización, datos y tecnología del software al completo. No es una metodología cerrada, es el acto de hacer algo para mejorar.

Ventajas de la modernización de una arquitectura del software

- **Los sistemas envejecen con mucha rapidez**. Vivimos en un mundo donde la tecnología cambia constantemente. Por ello, al vivir en un entorno tan cambiante, es muy normal que nuestra arquitectura se quede desfasada y debamos de modernizarla para ajustarlas a los cambios.
- **Modernizar nos permite escalar e innovar** a la hora de crear y mantener nuestros productos. Esto nos evita que el sistema sea un freno a la hora de querer expandir nuestro producto, o incluso realizar el propio mantenimiento.
- **Reduce drásticamente los costos y riesgos**. Esto se debe a que los gastos de mantenimiento se ven reducidos drásticamente al ser un sistema con una arquitectura moderna y actualizada. Lo mismo pasa con la deuda técnica, al no tomar decisiones precipitadas y cuidar nuestra arquitectura, nos ahorraremos muchos gastos en el futuro.

Desventajas de la modernización de una arquitectura del software

- **ROI** (retorno de inversión) es **muy difícil de cuantificar**.
- **Parón en la producción** o reducción drástica de la misma.
- Por lo normal, se considera **no prioritario** (“Si funciona, ¿para qué tocarlo?”).

Arquitectura moderna del software sociotécnica

El concepto de que **la arquitectura moderna del software es sociotécnica** significa que su diseño y éxito no dependen únicamente de factores técnicos (como tecnologías, patrones o infraestructura), sino también de **aspectos humanos, organizacionales y culturales**.

Es una visión integral que reconoce que los sistemas de software existen dentro de un contexto social y deben alinearse con las dinámicas de los equipos que los construyen, mantienen y usan.

A veces, hay distintas maneras de dividir la arquitectura, y tenemos que pensar cual de esas divisiones permite que los equipos trabajen de forma más independiente, sin bloquearse ni tener que coordinarse constantemente.

Criterios para saber qué modernizar

Identificar qué partes modernizar puede parecer un reto, pero basta con evaluar dos criterios clave:

- El **valor** que aporta la modernización.
- El **coste** que implica llevarla a cabo.

Cuantificar ambos aspectos permite estimar el **potencial de negocio** del cambio. Si el resultado es positivo, significa que la modernización tiene sentido estratégico. No todas las áreas deben modernizarse por obligación; es importante priorizar con base en el impacto real.

Además, al diseñar una nueva arquitectura, no se puede perder de vista la estructura de los equipos. **Negocio, equipos y arquitectura** conforman un triángulo interdependiente que sostiene un desarrollo eficaz y sostenible.

Herramientas de modernización

Si bien existen numerosas herramientas y enfoques para modernizar sistemas (como la Ley de Conway, event storming, Wardley Maps, Team Topologies o el análisis del comportamiento del código), no es necesario aplicarlas todas, pero sí es imprescindible responder a las **preguntas clave** que cada una plantea. Lo esencial es saber cuáles utilizar y en qué momento hacerlo, en función del problema específico que se busca resolver.

La clave para tomar buenas decisiones en un proceso de modernización es hacerse las preguntas adecuadas. Todo comienza por entender el **por qué**: cuál es el propósito del cambio. A partir de ahí se puede abordar el **cómo**, que implica definir la estrategia, clarificar el problema y planificar los primeros pasos. Pero este "cómo" solo puede construirse si primero se comprende con claridad el punto de partida (el estado actual del sistema y de la organización). Sin ese entendimiento, cualquier esfuerzo corre el riesgo de desviarse.

Relación entre datos y arquitectura

Hoy en día, los datos son fundamentales en cualquier sistema, pero a pesar de su evolución en volumen y variedad, muchas organizaciones siguen utilizando enfoques tradicionales de gestión, como metodologías en cascada, en contraste con los enfoques ágiles adoptados por otros equipos. Esto se complica aún más en procesos de migración de entornos **on-premise** a la nube, donde muchos intentan realizar un **lift and shift** sin adaptar sus sistemas a las nuevas tecnologías y entornos. Esto no solo genera problemas de rendimiento y costes, sino que también impide aprovechar las ventajas de la nube. Una forma de modernizar la gestión de datos es adoptar un enfoque como **data mesh**, que promueve principios como el pensamiento de producto aplicado a los datos, la federación por dominios y la gobernanza descentralizada, alineando mejor los datos con las necesidades del negocio y las metodologías ágiles.

Criterios concretos para saber si merece la pena modernizar

Hay ciertos **síntomas comunes** que indican que un sistema está frenando al negocio. Desde la perspectiva del negocio, añadir nuevas funcionalidades requiere más tiempo o esfuerzo del esperado, pequeñas mejoras pueden convertirse en meses de trabajo. Se rechazan ideas de valor porque "es arriesgado tocar ciertas cosas". Los errores o inconsistencias son frecuentes. En definitiva, **si el ralentiza la innovación y genera costes ocultos**, es hora de evaluar la modernización.

Desde la perspectiva de los datos, modernizar se vuelve crucial. **Cambios de filosofía** como pasar de on-premise a la nube o **cumplir con las normativas y leyes vigentes** (como el GDPR) es algo de **necesidad inmediata**.

Proyectos que no necesitan modernización

Hay sistemas que, gracias a su evolución natural, **se mantienen "vivos".** En estos casos, la **calidad** se incorpora día a día, a través de un enfoque constante en la **mejora continua** y **una cultura de refactorización** activa. La existencia de una fuerte **conciencia de calidad** a nivel organizativo, donde figuras como el CTO (Chief Technology Officer), con buen criterio que entiende el valor de estas prácticas y un CEO que confía en los equipos, impulsa esta visión.

Compatibilizar el trabajo de modernización con las demás prioridades del negocio

Una gran tensión en las organizaciones es encontrar espacio para **modernizar sin descuidar el desarrollo de nuevas funcionalidades**. La clave está en **planificar conscientemente**, presentar diferentes escenarios en una hoja de ruta:

- Uno con foco en modernización.
- Otro equilibrado.
- Uno centrado en producto.

Esto permite discutir con los stakeholders los trade-offs de cada uno, dar su opinión y decidir que es más valioso. Es decir, permite **conectar la modernización con los objetivos del negocio,** todo esto tiene que venir desde todas las capas de liderazgo, a través de una **comunicación clara y unificada** para que no haya dudas ni incoherencias.

Éxito y fracaso de modernizaciones

La experiencia muestra que **solo alrededor del 25% de los esfuerzos de modernización técnica tienen éxito completo**. Muchos otros fracasan o se quedan a medio camino, lo que puede resultar aún peor que no haber empezado.

Las causas más comunes de fracaso son:

- **Proyectos que se quedan a mitad de camino**.
- **Falta de una razón clara y sostenida** en el tiempo.
- Falta de capacidades técnicas del equipo para abordar problemas.
- La tentación a abandonar la modernización por resultados más visibles.

"Lo peor no es no modernizar. Lo peor es quedarse a medias y terminar con un sistema Frankenstein."