Software como una disciplina de ingeniería.

CURSO 2024-25

David Rodriguez Chanca – UO289521
Miguel Gutiérrez García - UO295650
Juan Fernández López - UO296143

¿Qué diferencias existen entre un ingeniero de software y un desarrollador de software?

- Chad dice que el desarrollo de software tiene la capacidad de poder desarrollar cosas que funcionen rápidamente, de manera que si queremos ver rápido una página web se lo pediríamos a un desarrollador de software. Pero en la ingeniería de software va un paso más allá de desarrollar algo que funcione, pues el producto tiene que ser riguroso, que cumpla objetivaos a largo plazo, escalables y mantenibles. Además un ingeniero de software siempre tiene diseñar un producto pensando en los posibles cambios que este puede sufrir en el futuro.
- Chad dice que esta mentalidad de lo que tiene que hacer el ingeniero de software la puedes leer en varios libros destacando el Code Complete de Steve McConnell, además Chad critica bastante el hecho de empezar un proyecto desarrollando software sin tener unas fases de diseño.

¿Que complejidades existen en el software?

- Chad destaca 3 complejidades en el software.
    - Objetiva: Esta trata de responder a las necesidades del cliente y no solo los requisitos que el cliente nos pide, por ejemplo: si un cliente nos pide una app para ventas, su objetivo real será aumentar las ventas a través de la app, por lo que si damos un producto que no lo cumple, poco le servirá al cliente.
    - Requisitos: Conflictos en las peticiones que nos hace los clientes, por ejemplo: si un cliente nos pide que una aplicación tenga un factor de doble autenticación y que también para logearse tiene que hacerse rápido. Como vemos ambos requisitos son válidos por separados pero juntos pueden generar conflictos, por lo que aquí tendríamos complejidad de requisitos.

- Solución: Una vez encontrada la solución para el problema, esta debe tener un diseño técnico que evite el caos, por ejemplo, si Netflix quisiera cambiar su algoritmo de recomendaciones, si tuviese un mal diseño obligaría a cambiar muchas cosas, mientras que con el diseño correcto solo se tendría que cambiar donde se necesita.

¿Qué es la “calidad institucionalizada” y por qué es esencial?

Chad menciona que no basta con que uno o dos especialistas se ocupen de la calidad: toda la organización debe comprometerse con ella. A esto se le llama “calidad institucionalizada”.

Si todas las personas (desde quienes recogen requisitos hasta quienes programan o prueban) no comparten esa visión, el riesgo de que aparezcan brechas en la calidad aumenta. El resultado son productos con más defectos y mayor costo de corrección.

Con la “calidad institucionalizada” se identifican y resuelven los problemas pronto, se fomenta la colaboración interdepartamental y se reduce la probabilidad de grandes retrasos al final. El ejemplo de Toyota y su filosofía de fabricar con calidad en cada fase se debería aplicar completamente al desarrollo de software.

La figura del “Chief Engineer”: un rol clave para integrar a todo el equipo

Chad sugiere que en cada equipo exista un rol con una visión global del proyecto, a menudo denominado “Chief Engineer”. Su misión es coordinar a los distintos perfiles (desarrolladores, testers, UX, negocio, etc.) y asegurar que todos sigan los mismos principios de arquitectura, calidad y coherencia.

Visión integral: Entiende cómo cada módulo del sistema se interrelaciona y se asegura de que las soluciones parciales encajen en el objetivo global.

Formación y mentoría: Al igual que un entrenador de fútbol, se encarga de transmitir buenas prácticas y estándares de diseño.

Guía para nuevos miembros: Facilita la incorporación de personal sin que se comprometa la calidad o se pierdan las reglas arquitectónicas.

De esta forma, el “Chief Engineer” no solo establece pautas, sino que promueve el crecimiento del equipo, manteniendo una calidad homogénea y permitiendo que el producto evolucione sin fisuras.

¿Cómo hacer un buen diseño/arquitectura del software?

Para hacer un buen diseño y arquitectura de software, primero es fundamental entender las necesidades y expectativas de los usuarios mediante la recopilación de requisitos a través de entrevistas, encuestas y análisis. Luego, se debe dividir el proyecto en módulos o componentes más pequeños, definiendo sus responsabilidades y cómo interactuarán entre sí, utilizando diagramas UML para visualizar la estructura y el flujo del sistema. Es importante aplicar principios de diseño como SOLID para crear un sistema robusto y mantenible, asegurándose de que el diseño sea modular, cohesivo y con bajo acoplamiento. Crear prototipos o maquetas del sistema ayuda a validar el diseño con los usuarios y obtener retroalimentación temprana, permitiendo identificar posibles problemas y realizar ajustes antes de la implementación. La documentación clara y detallada del diseño, incluyendo diagramas y descripciones de los módulos, facilita el mantenimiento y la evolución del software. Realizar revisiones periódicas del diseño con el equipo de desarrollo y otros stakeholders es clave para un diseño exitoso. En cuanto a la arquitectura de software, se debe definir la estructura general del sistema y seleccionar un estilo arquitectónico adecuado según las necesidades del proyecto, identificando los componentes principales y sus responsabilidades. Es esencial diseñar interfaces claras y consistentes para la comunicación entre componentes, considerando el rendimiento y la escalabilidad para manejar el crecimiento en la carga de trabajo. Incorporar medidas de seguridad y confiabilidad en la arquitectura, como la protección de datos y la gestión de accesos, es crucial. Realizar pruebas exhaustivas de la arquitectura y documentarla de manera detallada, incluyendo diagramas y decisiones arquitectónicas, facilita la comprensión y el mantenimiento del sistema a lo largo del tiempo. Seguir estos pasos te ayudará a crear un diseño y una arquitectura de software sólidos y eficientes.

BIBLIOGRAFIA

1 - Podcast – SE Radio 574: Chad Michel on Software as an Engineering Discipline

https://se-radio.net/2023/07/se-radio-574-chad-michel-on-software-as-an-engineering-discipline/

2- Don ´t panic labs (empresa en la que trabaja el entrevistado como rol de arquitecto) https://dontpaniclabs.com/

3 - El entrevistado lo cita como un libro donde se enumeran los principios de un buen diseño/arquitectura del software -> McConnell, Steve. *Code Complete*. 2nd ed., Microsoft Press, 2004.

4 - Libro del entrevistado, Chad Michel. Se le pregunta en varias ocasines por él. -> Durham, Doug, and Chad Michel. *Lean Software Systems Engineering for Developers: Managing Requirements, Complexity, Teams, and Change Like a Champ*. Apress, 2021.

5 – Libro sobre cómo hacer unos buenos requisitos.

Gerald Weinberg. Exploring Requirements: Quality Before Design. 1st ed.