Software as an Engineering discipline

Sara Inés Bolado – UO277494

Andrea Fuertes Carral – UO276299

Marcos González García - UO282587

1. Diferencia entre ingeniería software y desarrollo software

Chad Michel dice que desarrollar software es algo fácil, ya que puedes abrir un editor de código y ponerte a escribir, pero solo con eso no tienes garantía de que se cumplan los requisitos y de que realmente funcione. De esto es de lo que se encarga la ingeniería software. Hace una similitud diciendo que el desarrollo software es la parte 'científica' y la ingeniería es la parte 'aplicada', donde aplicas de la forma correcta estos conocimientos desarrollando sistemas reales con RIGOR, palabra en la que hace hincapié. La ingeniería software aporta la calidad al software.

2. Importancia del diseño y la planificación

Relaciona la resistencia a la idea de que se necesitan principios rigurosos de ingeniería en la industria software con el mundo ágil, diciendo que muchas veces se alega que las ideas 'no son ágiles', pero para tener un software que dure y tener agilidad en el proyecto, es necesario invertir tiempo en planificar. La metodología ágil, al no darle tanta importancia a la documentación, aporta una 'excusa' a muchos equipos para directamente ignorar esta parte, siendo fundamental. Ahora, las organizaciones tienen una idea en mente y quieren llevarla a cabo lo antes posible. El problema es que una vez que empiezas algo y tomas cierta dirección, es muy costoso cambiarle el rumbo, de ahí la importancia del diseño y la planificación, no ir directamente al código. Es mejor tener un plan imperfecto que no tener plan. Importante estimar tiempo y costes, también para hablar el lenguaje de las empresas.

3. Tres tipos de complejidad

Chad Michel habla de que existen tres tipos de complejidad: objetiva, de requisitos y de solución (cómo se estructura el software). Las herramientas ágiles han ayudado bastante en la industria a mejorar la complejidad de requisitos, al contrario que la de solución. La complejidad objetiva se refiere a los objetivos de los clientes. Muchas veces pensamos que nuestro objetivo es coger una lista de requisitos y llevarla a cabo, pero no tiene por qué corresponderse con lo que realmente quieren los clientes, de ahí la complejidad. Se debe alguna vez parar a pensar si realmente estamos tomando el camino correcto. Si queremos agilidad, necesitamos control sobre lo que está pasando, saber gestionar la complejidad. Es importante para las organizaciones aprender a sobrellevar los cambios de forma ágil, ser resolutivos, aunque haya cosas que cambiar se haga con control. No tener miedo al cambio, adaptar el sistema a estos sin tener que empezar de 0. La ingeniería software es hacer sistemas que sean mantenibles y puedan crecer.

4. Diseñar para el cambio: la clave de un Software sostenible

Chad enfatiza que, en el desarrollo de software, la clave para mantener sistemas escalables y manejables es diseñar pensando en el cambio. Compara este enfoque con cambiar un

calentador de agua: si el diseño es modular, el reemplazo es fácil y no afecta toda la estructura. El software debe seguir este principio, permitiendo modificaciones sin necesidad de rehacerlo desde cero. Un diseño modular facilita los cambios sin generar complicaciones mayores.

5. Diseño iterativo y el costo de la revisión tardía

Chad destaca que el diseño en software no es un proceso lineal, sino iterativo. Es fundamental contar con la retroalimentación de otros ingenieros para detectar problemas antes de que se conviertan en errores costosos. Jeff Doolittle compara el desarrollo de software con la construcción: los arquitectos diseñan con restricciones reales, y los ingenieros revisan la viabilidad de la estructura. En software, si el diseño no se ajusta durante el proceso, el sistema puede volverse insostenible, como si se estuviera rehaciendo una casa constantemente.

6. Refactorizacón y retrabrajo (rework)

La refactorización implica mejorar el código sin cambiar su comportamiento externo. Se hace con pruebas en su lugar y sin modificar abstracciones. El retrabajo involucra cambios mayores que afectan las pruebas y las abstracciones, y suele ser necesario cuando el diseño inicial no fue sólido. Chad menciona que un buen diseño desde el principio facilita la refactorización. Si el diseño es deficiente, el retrabajo es inevitable, lo que aumenta los costos y las complicaciones. Subraya que uno de los mayores costes en el desarrollo de software es el retrabajo. Si un error se detecta tarde, puede costar una gran cantidad de tiempo y esfuerzo corregirlo. Además, las soluciones rápidas (hotfixes) no resuelven los problemas a largo plazo. Si el código no se prueba correctamente, el equipo puede entrar en un ciclo donde un error resuelto genera otro, complicando aún más el proceso. La clave para evitar el retrabajo es que el diseño inicial permite la evolución sin necesidad de refactorizaciones muy grandes.

7. Calidad: Un Trabajo de Todos

Chad Michel también habla sobre la importancia de integrar la calidad en cada etapa del proceso, todos en el equipo deben ser responsables de ella. Menciona cómo Toyota, ya en sus orígenes, estableció en sus fábricas el Andon Cord, una cuerda de la que los trabajadores debían tirar si detectaban algún problema. Esta cuerda hacía que se detuviera la cadena de montaje para que todo el mundo colaborara en resolver el problema.

Explica después la diferencia entre control de calidad (evaluar el producto después de ser construido) y aseguramiento de calidad (integrar la calidad desde el inicio del proceso). Todos deben poder detectar problemas temprano antes de que se conviertan en fallos costosos.

8. Rol de Ingeniero Jefe

Chad Michel explica cómo el Ingeniero Jefe debe ser capaz de reconocer las fuerzas y debilidades de cada parte del equipo y debe ser capaz de adaptarse a ellas para organizarlo en todo momento, pues cambiará constantemente. Dicho en otras palabras, ver el proyecto y al equipo como un todo y tomar decisiones en torno a ello.

9. Expectativas de los Consumidores

Dicen que lo que se espera del software hoy en día no es comparable a lo que se dejaba pasar hace unos años en términos tanto de experiencia de usuario como de funcionalidad. Las expectativas de los usuarios crecen y crece con el paso del tiempo, haciendo que los requerimientos mínimos para cualquier proyecto dado aumenten con ellas.

10. Aplicar los principios de la Ingeniería del Software a nuestro contexto

Para terminar, Chad Michel explica que un buen inicio es buscar buenos líderes que valoren la ingeniería informática y se preocupen por enseñar correctamente a los nuevos integrantes del equipo. Indica que la empresa debe invertir tiempo en hacer que las nuevas incorporaciones aprendan y se acostumbren a la metodología interna.