Adaptive Software Development

¿Qué es?

Adaptive Software Development o Desarrollo de software adaptativo es una metodología de desarrollo de software, que se centra en la flexibilidad y la adaptación a los cambios que surgen durante el desarrollo de los proyectos. En lugar de seguir un plan rígido, ASD busca aprender constantemente, colaborar estrechamente y adaptarse a medida que el proyecto avanza.

Principios clave:

- Centrado en las personas: el desarrollo de software exitoso depende de la colaboración efectiva entre los miembros del equipo.
- Desarrollo iterativo: se divide el proyecto en ciclos pequeños e iterativos, lo que permite una mejora continua.
- Pruebas continuas: se integran las pruebas durante todo el proceso de desarrollo para garantizar la calidad.
- Aceptar el cambio:  reconoce que el cambio es inevitable y alienta a los equipos a adaptarse a los requisitos cambiantes y aprovechar el marco ágil.

Ciclo de vida del desarrollo de software adaptativo

El ciclo de esta metodología tiene tres fases principales, trabaja con iteraciones donde cada fase se alimenta de la anterior.

1) Especulación: Se enfoca en crear una visión integral del proyecto, se hace una suposición de qué se quiere lograr y cómo podría hacerse. Se establecen los objetivos y metas.  Es un momento de especulación de alto nivel, donde se esboza el panorama de posibilidades, dejando amplio margen para las tendencias del mercado y las necesidades de los clientes que darán forma a la obra maestra final. El equipo y cliente trabajan juntos para definir el alcance inicial.
2) Colaboración: Es donde se pone en práctica la teoría y enfatiza el trabajo colaborativo de los miembros del equipo. Se debe fomentar una cultura de comunicación abierta y aprendizaje continuo, y reiterar la importancia de la integración y las pruebas continuas.
3) Aprendizaje: Es el espacio de reflexión donde el equipo analiza lo que funcionó, lo que no y qué se aprendió. El equipo da un paso atrás para celebrar los éxitos, analizar los desafíos y trazar un rumbo de mejora para el siguiente ciclo. Esta cultura de aprendizaje impulsa la mejora continua, garantizando que, con cada iteración, los procesos se refinen, la toma de decisiones se agilice y el software se ajuste mejor a las necesidades del cliente.

¿Es gratuita o de paga?

Por ser una metodología no es una herramienta que se pueda comprar. No tiene licencias, solo es un conjunto de principios, ideas y prácticas que se pueden aplicar libremente en los proyectos, igual que otras metodologías como Scrum. Pero si podría incurrir en costos cuando se usan herramientas de software para aplicarlas como: Trello, Jira, etc. Cuando se contrata un consultor especializado en la metodología o en curso pagado (aunque no hay una certificación oficial obligatoria.)

Ventajas

- Centrado en los usuarios finales, lo que puede conducir a productos mejores y más intuitivos.
- Fomenta una mayor transparencia entre desarrolladores y clientes.
- Flexibilidad y adaptabilidad, ideal para proyectos con requisitos cambiantes. Permite modificar el rumbo sin rehacer todo desde cero.
- Tiempo de comercialización más rápido, el trabajo se entrega en ciclos cortos, lo que permite mostrar avances y obtener retroalimentación rápidamente.

Desventajas

- Exige una amplia participación del usuario, lo que puede ser difícil de facilitar.
- Integra pruebas en cada etapa, lo que puede aumentar los costos de un proyecto.
- Si no se gestiona bien, los ciclos repetitivos pueden agotar al equipo o al cliente.
- Se necesita un equipo disciplinado, autónomo y con buena comunicación para funcionar bien sin una estructura rígida.

Ejemplo: Desarrollo de aplicación de pedidos de comida para smartphones.

Especulación: Imaginar las funciones básicas que debería tener la app. Identificar las funcionalidades principales, como el registro de usuarios, la visualización del menú, la realización de pedidos y la integración de pagos. Luego, se hace un plan general (una hoja de ruta) pensando en cosas que podrían agregarse más adelante: Programa de puntos o recompensas, ofertas, etc.

Colaboración: Se crea una versión funcional básica con lo más esencial y se hacen pruebas con algunos usuarios para ver cómo reaccionan, y se escucha su opinión. El cliente y los usuarios participan activamente, ayudando a definir qué funciona bien y qué no.

Aprendizaje: Se analizan los datos y comentarios de los usuarios para identificar las funciones de mayor rendimiento y las áreas de mejora. No trata solo de corregir errores, sino de aprenden lo que realmente quiere el cliente.