DESARROLLO DE SOFTWARE

El desarrollo de software se refiere a un conjunto de actividades informáticas dedicadas al proceso de creación, diseño, despliegue y compatibilidad de software. El software en sí es el conjunto de instrucciones o programas que le dicen a una computadora qué hacer. Es independiente del hardware y hace que las computadoras sean programables.

El desarrollo de software es un proceso dinámico y desafiante que ha evolucionado significativamente a lo largo de los años para abordar la creciente complejidad de los proyectos tecnológicos. En sus primeras etapas, el proceso de desarrollo era lineal y rígido, con enfoques tradicionales que a menudo llevaban a dificultades en la adaptación a cambios y a la gestión de la incertidumbre.

Con el tiempo, se evidenció la necesidad de métodos más flexibles y ágiles, capaces de abordar los desafíos emergentes en el entorno tecnológico. La creciente demanda de sistemas más complejos y la rapidez con la que las tecnologías evolucionan han impulsado la búsqueda de enfoques que permitan una entrega más rápida y una mayor capacidad de adaptación.

En este contexto, surgen enfoques como el Proceso Unificado de Desarrollo (PUD) y el Desarrollo Iterativo, que buscan abordar las limitaciones de las metodologías más tradicionales. Estos enfoques reconocen la naturaleza dinámica del desarrollo de software, donde los requisitos pueden cambiar y evolucionar a lo largo del tiempo.

El Proceso Unificado de Desarrollo o PUD, es un marco de trabajo que se destaca por su enfoque iterativo e incremental. A través de sus distintas fases y disciplinas, el PUD busca ofrecer una estructura que permita gestionar eficientemente los desafíos inherentes al desarrollo de software, desde la concepción de la idea hasta la implementación y la transición del sistema.

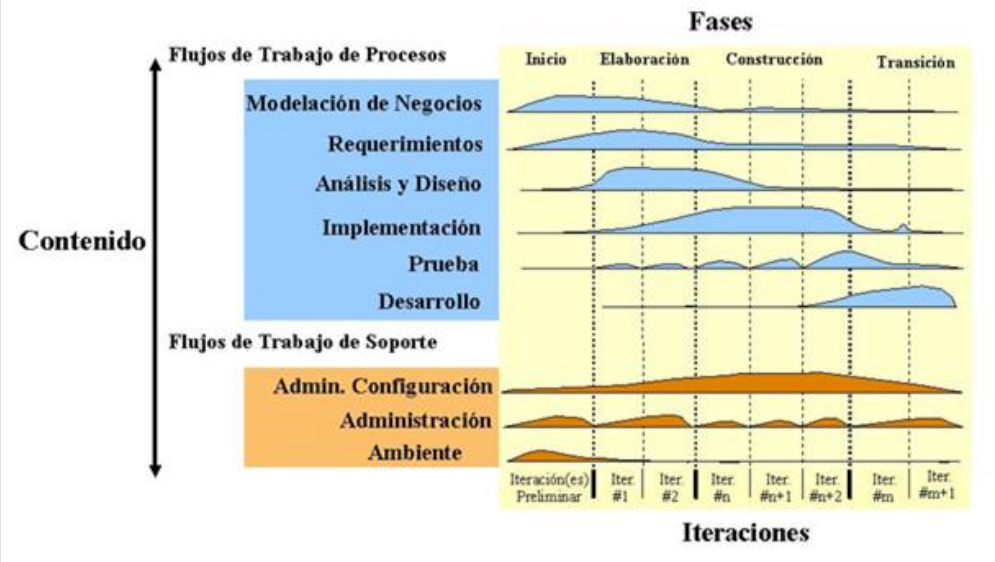
De manera complementaria, el Desarrollo Iterativo se ha convertido en un paradigma esencial en la ingeniería de software. La idea de construir y mejorar un sistema en ciclos repetitivos, con la capacidad de recibir y aplicar retroalimentación de manera continua, se alinea con la naturaleza cambiante de los requisitos y las expectativas de los usuarios finales.

Los proyectos de desarrollo de software enfrentan desafíos multifacéticos, que van desde la gestión efectiva de los requisitos hasta la entrega oportuna de productos de alta calidad. La necesidad de adaptabilidad se vuelve crucial en un

entorno donde la innovación y las demandas del mercado pueden cambiar rápidamente.

El Proceso Unificado de Desarrollo y el Desarrollo Iterativo han surgido como respuestas a estos desafíos, ofreciendo estructuras que se adaptan a la complejidad y la dinámica del desarrollo de software en la actualidad.

PROCESO UNIFICADO DE DESARROLLO



El Proceso Unificado de Desarrollo de Software o simplemente Proceso Unificado es un marco de desarrollo de software. No es simplemente un proceso, sino un marco de trabajo extensible que puede ser adaptado a organizaciones o proyectos específicos.

De la misma forma, el Proceso Unificado Racional, también es un marco de trabajo extensible, por lo que muchas veces resulta imposible decir si un refinamiento particular del proceso ha sido derivado del Proceso Unificado o del RUP. Por dicho motivo, los dos nombres suelen utilizarse para referirse a un mismo concepto.

CARACTERÍSTICAS PRINCIPALES

- **Dirigido por casos de uso:** En el Proceso Unificado los casos de uso se utilizan para capturar los requisitos funcionales y para definir los contenidos de las iteraciones. La idea es que cada iteración tome un conjunto de casos de uso o escenarios y desarrolle todo el camino a través

de las distintas disciplinas como diseño, implementación, prueba, etc.

- **Centrado en la arquitectura:** El Proceso Unificado asume que no existe un modelo único que cubra todos los aspectos del sistema. Por dicho motivo existen múltiples modelos y vistas que definen la arquitectura de software de un sistema. La analogía con la construcción es clara, cuando construyes un edificio existen diversos planos que incluyen los distintos servicios del mismo: electricidad, fontanería, etc.

- **Iterativo e incremental:** El Proceso Unificado es un marco de desarrollo iterativo e incremental compuesto de cuatro fases denominadas Inicio, Elaboración, Construcción y Transición. Cada una de estas fases es a su vez dividida en una serie de iteraciones (la de inicio puede incluir varias iteraciones en proyectos grandes). Estas iteraciones ofrecen como resultado un incremento del producto desarrollado que añade o mejora las funcionalidades del sistema en desarrollo.

  Cada una de estas iteraciones se divide a su vez en una serie de disciplinas que recuerdan a las definidas en el ciclo de vida clásico o en cascada: Análisis de requisitos, Diseño, Implementación y Prueba. Aunque todas las iteraciones suelen incluir trabajo en casi todas las disciplinas, el grado de esfuerzo dentro de cada una de ellas varía a lo largo del proyecto.

- **Enfocado en los riesgos:** El Proceso Unificado requiere que el equipo del proyecto se centre en identificar los riesgos críticos en una etapa temprana del ciclo de vida. Los resultados de cada iteración, en especial los de la fase de Elaboración deben ser seleccionados en un orden que asegure que los riesgos principales son considerados primero.

FASES DEL CICLO DE VIDA

1. **Fase de Inicio:** Tiene por finalidad definir la visión, los objetivos y el alcance del proyecto, tanto desde el punto de vista funcional como del técnico, obteniéndose como uno de los principales resultados una lista de los casos de uso y una lista de los factores de riesgo del proyecto. El principal esfuerzo está radicado en el Modelamiento del Negocio y el Análisis de Requerimientos. Es la única fase que no necesariamente culmina con una versión ejecutable.

2. **Fase de Elaboración:** Tiene como principal finalidad completar el análisis de los casos de uso y definir la arquitectura del sistema, además se obtiene

una aplicación ejecutable que responde a los casos de uso que la comprometen. A pesar de que se desarrolla a profundidad una parte del sistema, las decisiones sobre la arquitectura se hacen sobre la base de la comprensión del sistema completo y los requerimientos (funcionales y no funcionales) identificados de acuerdo al alcance definido.

3. **Fase de Construcción:** Está compuesta por un ciclo de varias iteraciones, en las cuales se van incorporando sucesivamente los casos de uso, de acuerdo a los factores de riesgo del proyecto. Este enfoque permite por ejemplo contar en forma temprana con versiones del sistema que satisfacen los principales casos de uso. Los cambios en los requerimientos no se incorporan hasta el inicio de la próxima iteración.

4. **Fase de Transición:** En esta fase final, el programa debe estar listo para ser probado, instalado y utilizado por el cliente sin ningún problema. Una vez finalizada esta fase, se debe comenzar a pensar en futuras novedades para la misma.

FLUJO DE TRABAJO

1. **Modelo de negocio:** Describe los procesos de negocio, identificando quiénes participan y las actividades que requieren automatización.

2. **Requerimientos:** Define qué es lo que el sistema debe hacer, para lo cual se identifican las funcionalidades requeridas y las restricciones que se imponen.

3. **Análisis y diseño:** Describe cómo el sistema será realizado a partir de la funcionalidad prevista y las restricciones impuestas (requerimientos), por lo que indica con precisión lo que se debe programar.

4. **Implementación:** Define cómo se organizan las clases y objetos en componentes, cuáles nodos se utilizarán y la ubicación en ellos de los componentes y la estructura de capas de la aplicación.

5. **Pruebas:** Busca los defectos a lo largo del ciclo de vida.

6. **Despliegue:** Produce el lanzamiento del producto y realiza actividades (empaque, instalación, asistencia a usuarios, etc.) para entregar el software a los usuarios finales.

7. **Administración del proyecto:** Involucra actividades con las que se busca producir un producto que satisfaga las necesidades de los clientes.

8. **Administración de configuración y cambios**: Describe cómo controlar los elementos producidos por todos los integrantes del equipo de proyecto en cuanto a: utilización/actualización concurrente de elementos, control de versiones, etc.

9. **Entorno:** Contiene actividades que describen los procesos y herramientas que soportan el equipo de trabajo del proyecto; así como el procedimiento para implementar el proceso en una organización.

ROLES

Dentro de esta metodología se presentan varios roles en el proceso de desarrollo y sus fases. Se agrupan en 5 grandes categorías, y cada una de ellas comprende varios sub-roles.

- **Analistas:** Como su nombre lo dice, analizan el proyecto, estableciendo los requerimientos, el sistema y el modelo del negocio. Son los encargados de definir el proyecto o producto y los recursos a utilizar.

- **Desarrolladores:** Este rol se encarga de diseñar y desarrollar el software, se comienza a trabajar en el proyecto y escribir el código utilizando la información previamente proporcionada por los analistas.

- **Gestores:** Son los encargados de administrar, supervisar y controlar el análisis, desarrollo y pruebas del proyecto. Estos deciden los cambios a implementarse en el software y la realización de pruebas.

- **Apoyo:** Sirven como utilidad para el desarrollo y complementación del proyecto de software.

- **Pruebas:** Categoría que comprende la verificación y prueba del software, declarando los puntos a reparar o mejorar.

- **Otros roles:** No son necesariamente importantes, pero toman un papel considerable en la planeación, desarrollo y distribución del software.
  - Stakeholders
  - Revisor
  - Coordinación de revisiones
  - Revisor técnico

VENTAJAS

- **Adaptabilidad a cambios**: Facilita la incorporación de cambios en los requisitos incluso en etapas avanzadas del desarrollo, así nos da mayor flexibilidad para ajustarse a las necesidades cambiantes del cliente o del entorno.

- **Gestión efectiva de riesgos**: La fase de elaboración se centra en la identificación y mitigación temprana de riesgos, mejorando la toma de decisiones y reduciendo la probabilidad de problemas significativos en etapas posteriores del desarrollo.

- **Visión clara del proyecto:** Facilita la planificación y la evaluación del progreso en comparación con los hitos establecidos porque cada fase tiene objetivos claros y entregas específicas.

DESARROLLO ITERATIVO

El desarrollo iterativo es una metodología de desarrollo de software en la que el proyecto se divide en pequeñas iteraciones o ciclos. Cada iteración es un proceso de planificación, diseño, implementación y pruebas que se realiza en un periodo de tiempo específico. Al final de cada iteración, se entrega el software para que pueda ser probado y evaluado por los usuarios.

La idea detrás del desarrollo iterativo es que es más fácil identificar y corregir los errores y problemas en etapas tempranas del proyecto, en lugar de descubrirlos al final. Cada iteración proporciona una oportunidad para que el equipo de desarrollo y el cliente revisen y ajusten los requisitos y objetivos del proyecto en función de lo aprendido en la iteración anterior. De esta manera, el proyecto se va adaptando y evoluciona según las necesidades.

PASOS

1. **Planificación y requisitos:** Durante este paso del proceso iterativo, se define el plan del proyecto que deberá estar alineado con los objetivos generales del proyecto. En esta etapa también se especificarán los requisitos esenciales que deben cumplirse para que el proyecto tenga éxito. Sin este paso, se corre el riesgo de iterar pero sin alcanzar los objetivos.

2. **Análisis y diseño:** Durante este paso, el equipo se centrará en las necesidades comerciales y los requisitos técnicos del proyecto. Si en el primer paso se han definido los objetivos, durante el segundo paso se llevará a cabo una lluvia de ideas para definir el diseño que eventualmente ayudarán a alcanzar esos objetivos.

3. **Implementación:** Durante el tercer paso, el equipo creará la primera iteración del entregable del proyecto. Esta iteración se basará en el análisis y diseño, y debería funcionar para alcanzar el objetivo final del proyecto. El nivel de detalle y el tiempo que se dedique a esta iteración dependerán del proyecto.

4. **Pruebas:** Ahora que hay una iteración, se probará de la forma que mejor funcione para el proyecto. Si es para mejorar una página web, por ejemplo, se realiza una prueba A/B para compararla con la página web actual. Si es para una función o producto nuevos, se realizarán pruebas de facilidad de uso con un grupo de clientes potenciales.

5. **Evaluación y revisión:** Después de las pruebas, el equipo puede evaluar el éxito de la iteración y centrarse en todo aquello que se necesite cambiar. ¿Esta iteración cumple con los objetivos del proyecto? ¿Por qué o por qué no? Si es necesario cambiar algo, se puede volver a iniciar el proceso

iterativo y repetir el paso dos para diseñar la iteración siguiente. Hay que tener en cuenta que la planificación y los objetivos iniciales deben ser los mismos para todas las iteraciones.

Si se vuelve a iniciar el proceso iterativo, hay que asegurarse de que todos estén alineados con los objetivos del proyecto. El proceso iterativo puede llevar semanas o meses, dependiendo de la cantidad de iteraciones que se realicen. Es importante centrarse en los objetivos del proyecto cada vez que se vuelva a iniciar el proceso iterativo, para no perder de vista la estrella guía.

VENTAJAS

- **Adaptabilidad a cambios:** Las iteraciones permiten ajustes continuos en función de los cambios en los requisitos del cliente o del mercado, así el equipo puede responder rápidamente a nuevas necesidades o aclaraciones, mejorando la capacidad de adaptación del proyecto.

- **Retroalimentación continua:** Cada iteración proporciona oportunidades para recibir retroalimentación del cliente o de los usuarios finales y facilita la mejora constante del producto, asegurando que esté alineado con las expectativas y requisitos reales.

- **Mejora continua de calidad:** Cada iteración puede incluir actividades de prueba y corrección de errores, contribuyendo a la mejora continua de la calidad del software, minimizando la acumulación de defectos y mejorando la estabilidad del producto final.

COMPARACIÓN

	PROCESO UNIFICADO DE DESARROLLO	DESARROLLO ITERATIVO
Estructura y Organización	Está organizado en fases claramente definidas y cada fase tiene disciplinas específicas.	Está organizado en ciclos repetitivos, hay mayor flexibilidad en la ejecución de disciplinas en cada iteración.
Planificación y Gestión del Proyecto	La planificación debe ser detallada al principio del proyecto. Nos entrega productos	Hay una planificación a corto plazo que se ajusta al final de la iteración. Nos entrega

	significativos al final de cada fase.	incrementos funcionales en cada iteración.
Gestión de Cambios y Adaptabilidad	Puede adaptarse, pero los cambios significativos se consideran en las fases iniciales.	Se adapta fácilmente a cambios en los requisitos a lo largo de las iteraciones.
Gestión de Riesgos	Está enfocado en la identificación y mitigación de riesgos en las primeras fases.	Aborda y mitiga riesgos a lo largo de las iteraciones.
Documentación y Artefactos	La documentación debe ser detallada en cada fase.	La documentación es necesaria pero más enfocada en la entrega incremental.
Colaboración y Comunicación	Roles claramente definidos, con interacción entre ellos.	Promueve la colaboración continua entre el equipo y los stakeholders.
Enfoque al cliente	Retroalimentación del cliente principalmente en fases finales.	Busca retroalimentación del cliente en cada iteración.
Flexibilidad y Agilidad	Más rígido en términos de cambios durante el desarrollo.	Mayor capacidad de adaptación y respuesta a cambios.
Aplicabilidad a Proyectos	Puede ser más adecuado para proyectos grandes y complejos.	Especialmente útil en proyectos donde los requisitos son propensos a cambios.

CONCLUSIÓN

Ambos modelos de Desarrollo de Software aportan perspectivas únicas y se adaptan a diferentes y necesidades dentro del amplio espectro de proyectos tecnológicos.

El PUD destaca la importancia de la planificación detallada y la mitigación de riesgos en las primeras etapas. Este enfoque resulta eficaz en proyectos grandes y complejos, donde la estabilidad y la estructura son esenciales.

En contraste, el Desarrollo Iterativo se destaca por su agilidad inherente y su capacidad para adaptarse a cambios continuos. Las iteraciones ofrecen entregas incrementales y retroalimentación continua, promoviendo una respuesta rápida a las variaciones en los requisitos y prioridades del cliente. Este enfoque encuentra su fuerza en proyectos dinámicos y cambiantes, donde la flexibilidad y la adaptabilidad son críticas.

La elección entre el PUD y el Desarrollo Iterativo dependerá de las necesidades y requisitos únicos de cada proyecto. Puede ser beneficioso combinar elementos de ambos modelos para aprovechar las fortalezas de cada uno, construyendo así un

marco de desarrollo personalizado que se ajuste mejor a las necesidades particulares del equipo y del cliente.

Tanto el PUD como el Desarrollo Iterativo ofrecen herramientas valiosas para los equipos de desarrollo, y su aplicación reflexiva puede llevar a la entrega exitosa de productos de software que cumplan y superen las expectativas del cliente.

REFERENCIAS

https://www.ecured.cu/Proceso_unificado_de_desarrollo#Caracter.C3.ADsticas_Principales_de_RUP

https://es.wikipedia.org/wiki/Proceso_unificado

https://sentrio.io/blog/diferencias-entre-desarrollo-iterativo-e-incremental/#%C2%BFQue_es_el_desarrollo_iterativo

https://www.ibm.com/docs/es/engineering-lifecycle-management-suite/lifecycle-management/6.0.3?topic=scenarios-iterative-development

https://blog.hubspot.es/website/diseno-web-iterativo

https://es.smartsheet.com/iterative-process-guide

https://blog.comparasoftware.com/ciclo-de-vida-iterativo-que-es-y-cuales-son-sus-ventajas/

https://www.nimblework.com/es/agile/desarrollo-iterativo-e-incremental/

https://www.ibm.com/es-es/topics/software-development

https://es.wikipedia.org/wiki/Ingenier%C3%ADa_de_software

https://asana.com/es/resources/iterative-process