Desarrollo de contenido

IU Digital de Antioquia
INSTITUCIÓN UNIVERSITARIA DIGITAL DE ANTIOQUIA

Unidad 1

Introducción a la Ingeniería de Software y Datos

Ingeniería de Software y Datos

ORD No 74 de 2017 · VIGILADA MinEducación

Introducción al Desarrollo de Software

Continuamente están apareciendo nuevas tecnologías que buscan satisfacer diferentes necesidades, y el software es una parte importante. Si miramos a nuestro alrededor, la mayoría de los dispositivos electrónicos usan software, algunos de forma más especializada que otros, nuestros teléfonos celulares poseen múltiples aplicaciones, los televisores tienen opciones de menú avanzadas, incluso tecnologías como la domótica ya hacen posible controlar una vivienda de forma remota. La tecnología continúa avanzando y con ella la necesidad de crear herramientas de software para optimizar, y dar soporte a herramientas existentes.

Por medio de esta asignatura se busca comprender el fundamento de la ingeniería de software y datos, el alcance, competencias, retos y dificultades de los estudiantes en el campo de acción.

Te invitamos a conocer más sobre el curso de Introducción a la Ingeniería de Software y Datos, observando el video de presentación, leyendo los objetivos y explorando el mapa del curso para que descubras sus contenidos.

Objetivo general

Realizar un acercamiento de la Ingeniería desde sus inicios en la historia, permitiendo al estudiante tener un panorama claro y completo frente al origen de la Ingeniería de Software y la evolución que ha tenido desde los tiempos hasta la actualidad desde un enfoque de analítica de datos.

Objetivos específicos

- Reconocer los elementos básicos de la ingeniería de software y datos.
- Identificar los aspectos determinantes de los estudiantes en el campo de acción.
- Conocer los retos y dificultades del campo de acción del desarrollo de software.
- Describir los principios y conceptos fundamentales en la profesión.

Mapa del curso

Unidad 1. Introducción al desarrollo del software

La construcción de herramientas de software requiere de un proceso que busca satisfacer las necesidades de los clientes por medio de las herramientas desarrolladas.

En esta unidad se darán a conocer la definición, el concepto de desarrollo de software y los tipos de software existentes. También se abordarán los fundamentos de software, características, cualidades, ciclo de vida de ese desarrollo y algunas metodologías de creación; buscando que el estudiante adquiera un conocimiento introductorio sobre este campo profesional.

En la Unidad 1, Introducción al Desarrollo del Software, estudiaremos cuatro saberes esenciales que estarán asociados a las Actividades de Aprendizaje, las cuales están diseñadas para fortalecer los conocimientos adquiridos en tu proceso de formación.

- Definición y conceptos.
- Software como fundamento.
- Características del software.
- Metodologías de desarrollo de software.

!Te damos la bienvenida a la Unidad 1!

Tema 1. Definición y conceptos

A continuación, te presentamos la definición y conceptos de desarrollo del software.

¿Qué es software?

Comúnmente se conoce como la parte lógica e intangible de un computador, pero siendo más específicos, software son todas aquellas rutinas, instrucciones o aplicaciones informáticas que son ejecutadas de manera ordenada para llevar a cabo diferentes tareas o actividades en los dispositivos.

En los últimos años se han experimentado grandes cambios en el área de la computación, que han llevado a la generación de nuevas tecnologías, metodologías y enfoques de desarrollo que han repercutido en las organizaciones y en la forma de hacer y ejecutar software (Delgado, n.d.).

El desarrollo de software es un proceso que consta de diferentes fases, las cuales se denominan Ciclo de vida del software y que serán explicadas posteriormente con mayor detalle. Un software puede ser desarrollado no solo para computadores, sino también para cualquier dispositivo que tenga capacidad de procesamiento, como por ejemplo teléfonos móviles, relojes inteligentes, televisores, vehículos, etc.

Las **aplicaciones de software** se construyen haciendo uso de lenguajes de programación, los cuales son un conjunto de reglas sintácticas y semánticas utilizadas para codificar las instrucciones que debe seguir un programa al ser ejecutado. Las instrucciones escritas sobre un lenguaje de programación son traducidas a lenguaje de máquina para que los computadores puedan interpretarlas; este proceso se lleva a cabo por medio de unas herramientas llamadas compiladores.

Tipos de software

Existen tres grandes tipos de software los cuales son: software del sistema, software de programación y software de aplicaciones.

Software del sistema

Son programas que interactúan directamente entre los usuarios y los componentes de hardware del computador; algunos de estos programas sirven como base para que sobre ellos corran otro tipo de aplicaciones. Dentro de esta categoría podemos encontrar sistemas operativos como Windows, Linux, Android, Mac OS, y controladores de dispositivos y servidores.

Software de programación

Son programas que permiten a los desarrolladores crear nuevas herramientas de software; a este grupo pertenecen, por ejemplo, los lenguajes de programación, compiladores y depuradores.

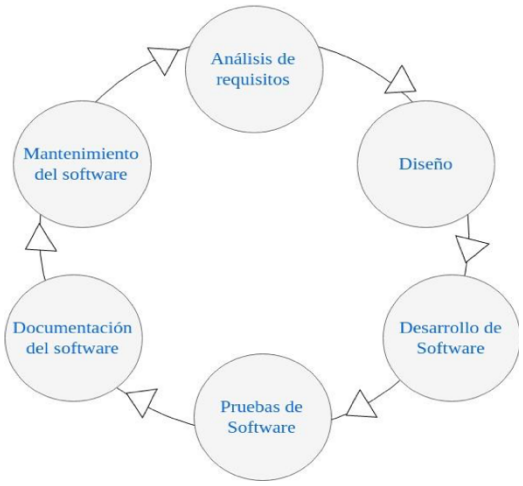
Software de aplicaciones

Son todos los programas creados para que los usuarios realicen tareas específicas; a esta categoría pertenecen la mayoría de las aplicaciones que conocemos y que utilizamos en nuestros diferentes dispositivos. Algunos ejemplos de este grupo son:

- Procesadores de texto y hojas de cálculo como: Word, Excel, Bloc de notas.
- Software empresarial, software contable, sistemas de inventario, sistemas de facturación.
- Plataformas web, redes sociales, tiendas virtuales, sistemas de reservas, catálogos, sitios informativos.
- Videojuegos.
- Clientes de correo electrónico.
- Software a la medida.
- App para dispositivos móviles.
- Aplicaciones con Inteligencia Artificial

Tema 2. Software como fundamento

Para llevar a cabo el proceso de desarrollo del software, se deben considerar diferentes etapas, las cuales permitirán al equipo de desarrollo estructurar adecuadamente el producto, y enfocar exitosamente su construcción. A estas etapas del desarrollo de software también se les llaman ciclo de vida del software, y pueden variar dependiendo del modelo de desarrollo que utilice cada organización.



Las etapas comunes que podemos encontrar en los diferentes modelos de desarrollo de software son:

- Etapa de análisis de requisitos.
- Etapa de diseño.
- Etapa de desarrollo.
- Etapa de pruebas o verificación.
- Etapa de documentación.
- Etapa de mantenimiento.

El orden de desarrollo de estas etapas depende del modelo que utilicen las empresas para el desarrollo del software.

Tema 3: Etapas de desarrollo de software

Como hemos visto, el desarrollo de software se lleva a cabo en varias etapas, las cuales dependen del modelo que cada empresa desarrolladora de software adopte.

Las etapas más comunes que podemos encontrar en los diferentes modelos de desarrollo de software son:

Etapa de análisis de requisitos

Consiste en identificar las verdaderas necesidades que tienen los clientes. Para llevar a cabo esta etapa, se requiere de un equipo que posea el conocimiento y la experiencia necesarios para realizar el proceso e identificar los requerimientos, ya sean incompletos,

ambiguos o contradictorios, porque en la mayoría de los casos los clientes no tienen esta información lo suficientemente clara.

Es preciso reconocer y abordar a los stakeholder (interesados), que son las personas o recursos, que de una u otra manera tendrán interacción con la herramienta, para obtener de ellos la información relevante que ayude en la creación de la herramienta, o para validar los requisitos previamente identificados.

Los requisitos obtenidos deben ser correctamente documentados, según el modelo de desarrollo con el cual se esté trabajando. Para algunos modelos se generará un documento de especificación de requisitos y, para otros modelos se documentará la información por medio de historias de usuario.

Etapa de diseño

En esta etapa se describe el sistema sin llegar a ser específico en su desarrollo; se identifican las características técnicas del software, las cuales permitirán implementarlo adecuadamente; se define la estructura general que tendrá el sistema, la cual se denomina el diseño arquitectónico con el cual se busca satisfacer los atributos de calidad del sistema; se realiza un análisis de la tecnología que se debe utilizar, como por ejemplo la infraestructura sobre la cual se ejecutará la herramienta, y las condiciones de red necesarias o dispositivos adicionales que se requieren para su utilización. .

El diseño arquitectónico del software juega un papel fundamental para guiar el desarrollo en esta etapa. Una de las múltiples estructuras que la componen, se enfoca en dividir el sistema en componentes que serán desarrollados por individuos o grupos de individuos. La identificación de esta estructura de asignación de trabajo es esencial para apoyar las tareas de planeación del proyecto (Cervantes, 2018).

Etapa de desarrollo

En esta etapa se lleva a cabo el desarrollo de la herramienta con el lenguaje de programación definido en la etapa de diseño; los requisitos y el diseño de la herramienta previamente realizados, se convierten en código que puede ser interpretado por los dispositivos apropiados, y ejecutado por los usuarios.

La correcta ejecución de las etapas anteriores, facilitan la labor para llevar a cabo la etapa de desarrollo; si por el contrario, las etapas anteriores tuvieron una implementación deficiente, en este punto se tendrán reprocesos que implicarán un aumento en el tiempo de desarrollo, y de costos o variaciones en el alcance de la herramienta.

Etapa de prueba o verificación

Consiste en comprobar que el software realiza de manera adecuada las tareas que se especifican en las etapas de análisis, de requisitos y de diseño. Se recomienda iniciar la realización de las pruebas desde la etapa de desarrollo, porque esto permitirá identificar problemas de forma temprana y dar solución a estos. Realizar ajustes o correcciones de errores en etapas avanzadas del desarrollo de software, puede generar grandes reprocesos que implican aumento de los costos de desarrollo, del tiempo establecido o/y del alcance de la herramienta.

La etapa de pruebas o verificación está relacionada directamente con el aseguramiento de la calidad del software, lo cual representa un campo de acción dentro del área de desarrollo de éste.

Etapa de documentación

Esta etapa se lleva a cabo de forma transversal y se refiere específicamente a la generación de la documentación total del proyecto, como los documentos de requisitos de software, de diseño y arquitectura, los diagramas de bases de datos, los de pruebas, los manuales de usuario, el código fuente y toda aquella información que pueda ser de importancia para la gestión, el mantenimiento y las futuras actualizaciones del sistema.

Etapa de mantenimiento

Esta etapa se lleva a cabo después que el software ha sido puesto en producción; **el mantenimiento puede ser correctivo**, en el cual se resuelven inconvenientes que no fueron identificados en la fase de pruebas, o puede ser de **tipo evolutivo**, en el cual se mejora la funcionalidad del sistema o se lleva a cabo el desarrollo de nuevos requisitos.

En este último punto es necesario tener muy claro el alcance inicial del sistema, debido a que puede ocurrir que las nuevas funcionalidades solicitadas por el cliente ameriten realizar cobros adicionales (según la forma de contratación definida).

Cada vez que se acuerde llevar a cabo el desarrollo de nuevas funcionalidades, es necesario reiniciar las diferentes etapas de desarrollo. Por este motivo, se considera un proceso cíclico.

Tema 4: Etapas de desarrollo de software

Es necesario tener claras las principales características que el software debe tener cuando se está desarrollando. Estas características han de ser identificadas al momento de realizar el análisis de requisitos y el diseño de la herramienta.

Dependiendo del tipo de software que se desarrolló, implica mayor o menor prioridad para determinadas características.

Analiza los siguientes ejemplos para mayor comprensión:

Desarrollar un software que soporte las diferentes operaciones que se realizan en un banco:
Para este tipo de aplicación es probable que una de las características que se debe tener en cuenta es la confiabilidad porque un error en la plataforma o un comportamiento indeseado, pueden hacer que las personas no utilicen la herramienta, incluso puede llegar a disminuir la confianza que tienen los clientes en la entidad.

Desarrollar un software para soportar una tienda virtual:
Para el dueño de la tienda en línea puede ser de gran importancia que sus clientes encuentren fácilmente sus productos y realicen la compra de forma sencilla. En este caso, el sistema debe ser amigable, con el fin de facilitar la compra de productos.

Características de un software

Usabilidad	Que sea fácil de operar.
Flexibilidad	Que sea fácil de modificar por los desarrolladores.
Portabilidad	Que pueda ser utilizado en diferentes dispositivos.
Interoperabilidad	Debe poseer la capacidad de compartir información con otras aplicaciones fácilmente.
Integridad	Al ejecutar el software no debe causar efectos secundarios.
Fiabilidad	El software no debe tener defectos ni fallar durante la ejecución.
Eficiencia	El software debe hacer uso eficaz de los recursos disponibles, como memoria, espacio de almacenamiento, procesamiento, etc.
Reutilización	El código fuente del software debe poderse utilizar en otras aplicaciones.
Mantenibilidad	El mantenimiento del software debe ser fácil de realizar, y poderse llevar a cabo por cualquier usuario.
Extensibilidad	Se deben poder crear fácilmente nuevas funcionalidades.
Escalabilidad	Debe adaptarse al aumento de trabajo fácilmente.
Capacidad de pruebas de software	Debe ser fácil realizar pruebas a la herramienta.
Modularidad	Debe estar creado por unidades o módulos independientes.

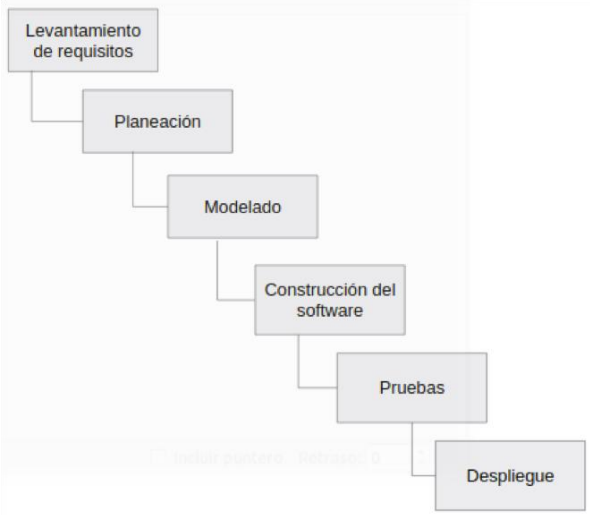
Tema 5. Metodologías de desarrollo de software

Se definen las metodologías de software, como los marcos que permiten la estructuración, planificación y control de los proyectos de software (Medina Velandia, 2015).

La implementación de estas metodologías brinda a la organización y al equipo de trabajo, un marco que permite construir las aplicaciones de manera eficiente y ordenada, y controlar el avance del proyecto, procurando siempre la obtención de un producto acorde a las necesidades del cliente y al presupuesto estimado.

Existen varias metodologías de desarrollo de software:

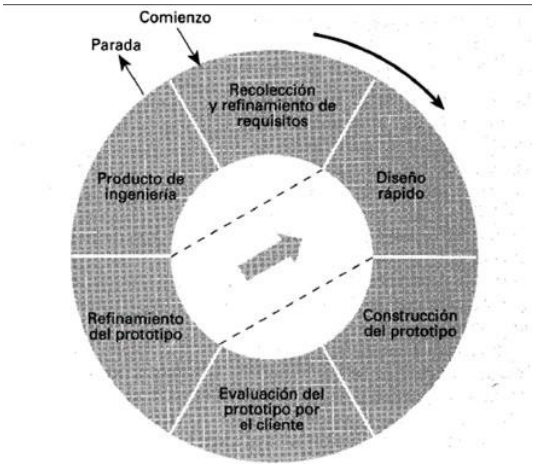
Modelo cascada



También es llamado ciclo de vida clásico, este modelo ejecuta las diferentes etapas de desarrollo de forma secuencial, iniciando por el levantamiento de requisitos y continuando ordenadamente con planeación, modelado, construcción del software, pruebas y despliegue. El modelo en cascada se desarrolló tomando como base los modelos de producción de productos.

Este tipo de modelo se utiliza especialmente en proyectos donde los requisitos son muy estables, no varían durante el proceso de desarrollo. Esta característica no se presenta de manera recurrente.

Prototipado

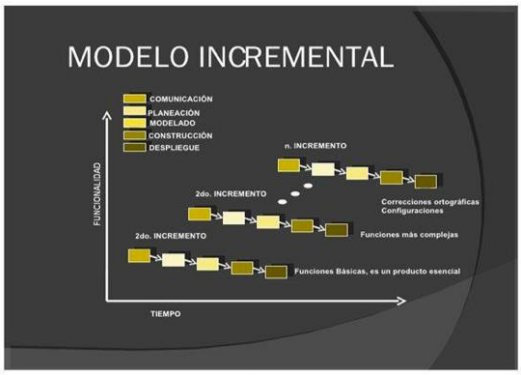


Esta metodología permite que empresas de desarrollo presenten a sus clientes prototipos de la solución que ayuden a validar la funcionalidad de la herramienta, los requisitos previamente obtenidos y realizar cambios sobre estos antes de construir la solución definitiva.

Importante

Es importante tener en cuenta que la construcción del prototipo debe ser rápida y permitir destruirse para luego crear nuevos prototipos basados en la realimentación que el cliente proporcione. Los desarrolladores deben evitar partir del prototipo para realizar el desarrollo de la herramienta, porque al desarrollarlos de forma rápida no se tienen en cuenta estándares de calidad.

Modelo incremental



Modelo Incremental

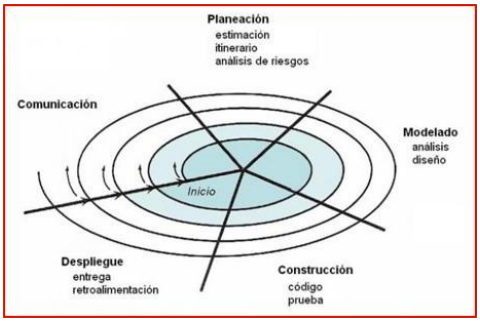
Este modelo combina el desarrollo en cascada con el modelo de construcción de prototipos. Consiste en ir generando incrementos de la funcionalidad del software haciendo entregas de funcionalidades específicas de la plataforma, conforme transcurre el tiempo.

Los incrementos pueden ser funcionalidades incompletas, pero dan al cliente la oportunidad de conocer el avance del proyecto y evaluar las entregas realizadas.

Importante

El modelo de desarrollo continúa siendo lineal; en cada incremento se ejecutan todas las etapas de desarrollo de software, presentes en el modelo en cascada.

Modelo en espiral



El modelo fue creado por Barry Boehm; él lo define como “un generador de modelo de proceso, impulsado por el riesgo, que se usa para guiar la ingeniería concurrente con participantes múltiples de sistemas intensivos en software.

Tiene dos características distintivas principales:

1. Enfoque cíclico para el crecimiento incremental del grado de definición de un sistema y su implementación, mientras que disminuye su grado de riesgo.
2. Es un conjunto de puntos de referencia de anclaje puntual para asegurar el compromiso del participante con soluciones factibles y mutuamente satisfactorias" (Pressman et al., 2010).

Tema 6. Metodologías ágiles

En febrero de 2001, tras una reunión celebrada en Utah, EEUU, nace el término "ágil" aplicado al desarrollo de software. En esta reunión participaron diecisiete expertos de la industria del software, incluyendo algunos de los creadores o impulsores de metodologías de software. Su objetivo se basó en esbozar los valores y principios que deberían permitir a los equipos, desarrollar software rápidamente y responder a los cambios que pudieran surgir a lo largo del proyecto.

Se pretendía ofrecer una alternativa a los procesos de desarrollo de software tradicionales, caracterizados por ser rígidos y dirigidos por la documentación que se genera en cada una de las actividades desarrolladas (Repositorio institucional de la Universidad de Las Tunas, 2012).

A partir de esta reunión surge **"Agile Alliance",** organización sin ánimo de lucro dedicada a promover los conceptos relacionados con el desarrollo ágil de software y acompañar a las organizaciones para que adopten dichos conceptos. Como punto de partida o base fundamental de las metodologías ágiles, se redactó y proclamó el manifiesto ágil (Herrera Uribe &Valencia Ayala, 2007).

Este manifiesto hace énfasis en cuatro valores principales que debe soportar el desarrollo de software.

- Individuos e interacciones sobre procesos y herramientas.
- Software funcionando sobre documentación extensiva.
- Colaboración con el cliente sobre negociación contractual.
- Respuesta ante el cambio sobre seguir un plan.

Y se sustenta en 12 principios (Repositorio institucional de la Universidad de Las Tunas, 2012).

1. La prioridad es satisfacer al cliente mediante tempranas y continuas entregas de software que le aporten un valor.
2. Dar la bienvenida a los cambios. Se capturan los cambios para que el cliente tenga una ventaja competitiva.
3. Entregar frecuentemente software que funcione desde un par de semanas a un parde meses, con el menor intervalo de tiempo posible entre entregas.
4. La gente del negocio y los desarrolladores deben trabajar juntos a lo largo del proyecto.
5. Construir el proyecto en torno a individuos motivados. Darles el entorno y el apoyo que necesitan y confiar en ellos para conseguir finalizar el trabajo.
6. El diálogo cara a cara es el método más eficiente y efectivo para comunicar información dentro de un equipo de desarrollo.
7. El software que funciona es la medida principal de progreso.
8. Los procesos ágiles promueven un desarrollo sostenible. Los promotores, desarrolladores y usuarios deberían ser capaces de mantener una paz constante.
9. La atención continua a la calidad técnica y al buen diseño mejora la agilidad.
10. La simplicidad es esencial.
11. Las mejores arquitecturas, requisitos y diseños surgen de los equipos organizados por sí mismos.
12. En intervalos regulares, el equipo reflexiona respecto a cómo llegar a ser más efectivo y, según esto, ajustar su comportamiento.

Este tipo de metodologías trae para las organizaciones y para los clientes varias ventas como son:

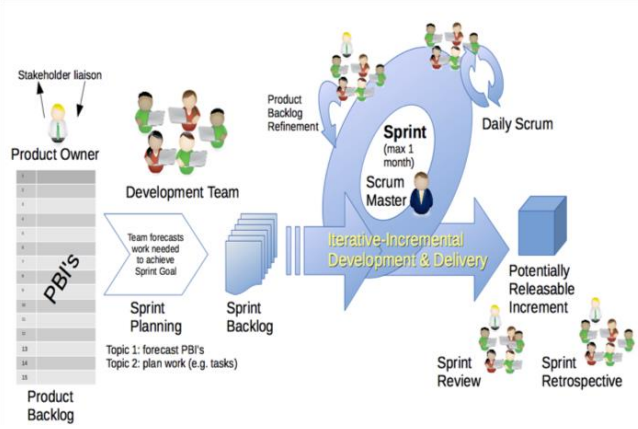
- Se mejora la satisfacción del cliente. El cliente está siempre involucrado en el desarrollo del proyecto y puede ver avances significativos, conocer los inconvenientes que se puedan presentar e intervenir en su solución.
- Rápida respuesta al cambio. Al realizar entregas continuas sobre el producto, se pueden detectar inconvenientes o se pueden generar cambios en los

requerimientos los cuales pueden ser abordados rápidamente, lo cual resulta menos costoso que hacerlo cuando el proyecto está finalizado.

- Se mejora la motivación del equipo de trabajo. Este puede conocer completamente el estado del proyecto y dar aportes valiosos a la solución de los problemas.
- Entregas parciales del producto. Este tipo de metodologías permite realizar entregas funcionales del producto durante el desarrollo del proyecto, lo cual genera ventajas para el cliente al no tener que esperar hasta la finalización total del proyecto para poder comenzar a utilizarlo.
- Mejora en la calidad del producto. El tener una interacción directa con el cliente, genera que el producto final esté acorde a sus necesidades.
- Eliminación de tareas innecesarias. En la mayoría de los proyectos donde se realiza un levantamiento general de requisitos del proyecto, se desarrollan módulos o funcionalidades que al final del proyecto no son utilizadas; este tipo de metodologías permite priorizar los elementos a desarrollar, excluyendo aquellos que no ofrecen valor a la organización.

A continuación, nombraremos algunas de las metodologías ágiles más comunes:

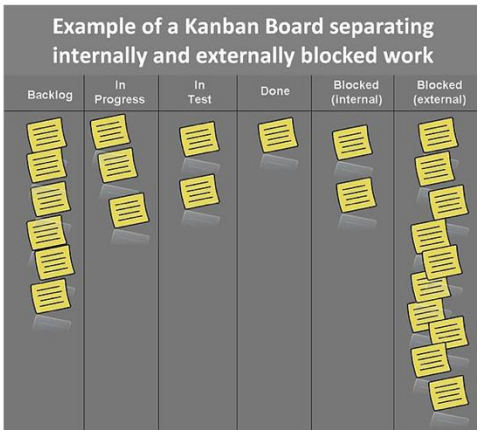
Scrum:



Es un marco de trabajo que promueve la colaboración y la comunicación entre los integrantes del equipo de trabajo; esta metodología promueve la auto organización.

Para tener claro cómo se lleva a cabo la metodología, debes revisar el documento sobre metodologías ágiles.

Kanban:

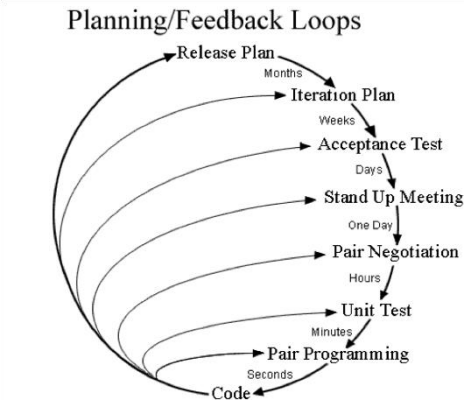


Es una técnica creada por Toyota para controlar el avance del trabajo en la creación de automóviles, pero ha sido implementada ampliamente en el desarrollo de proyectos de software, apoyando a su vez metodologías como Scrum.

Esta técnica tiene como regla visualizar siempre las fases del ciclo de producción o el flujo de trabajo y se caracteriza por ser muy visual, debido a que, en esta,
se utiliza un tablero en el cual se ingresan todas las tareas que se deben realizar. El tablero está dividido por estados, y cada tarea se va desplazando a un estado diferente a medida que se va desarrollando.

Este tipo de metodología permite visualizar claramente el estado del proyecto y las actividades que cada persona tiene asignadas. Para su implementación se deben tener en cuenta aspectos como: la edición del tiempo de desarrollo de las tareas y cantidad máxima de tareas por estado.

Extreme Programming:



Es una metodología orientada a aplicar mejores prácticas de ingeniería de software en el desarrollo de proyectos, mejorar la productividad en los proyectos y aumentar la calidad del desarrollo de los productos de software.

En esta metodología se difunden cinco valores y se aplican diferentes prácticas: comunicación, simplicidad, retroalimentación, coraje y respeto.

Esta licencia permite a otros distribuir, remezclar, retocar, y crear a partir de esta obra de manera no comercial y, a pesar que sus nuevas obras deben siempre mencionar a la IU Digital y mantenerse sin fines comerciales, no están obligados a licenciar obras derivadas bajo las mismas condiciones.

IU Digital de Antioquia
INSTITUCIÓN UNIVERSITARIA DIGITAL DE ANTIOQUIA

ORD No 74 de 2017 · VIGILADA MinEducación

www.iudigital.edu.co