UNIVERSIDAD UCATEC

ANALISIS Y DESARROLLO DE SISTEMAS I

Modelos de Ciclos de Vida de los Sistemas

ING. CLAUDIA LOPEZ

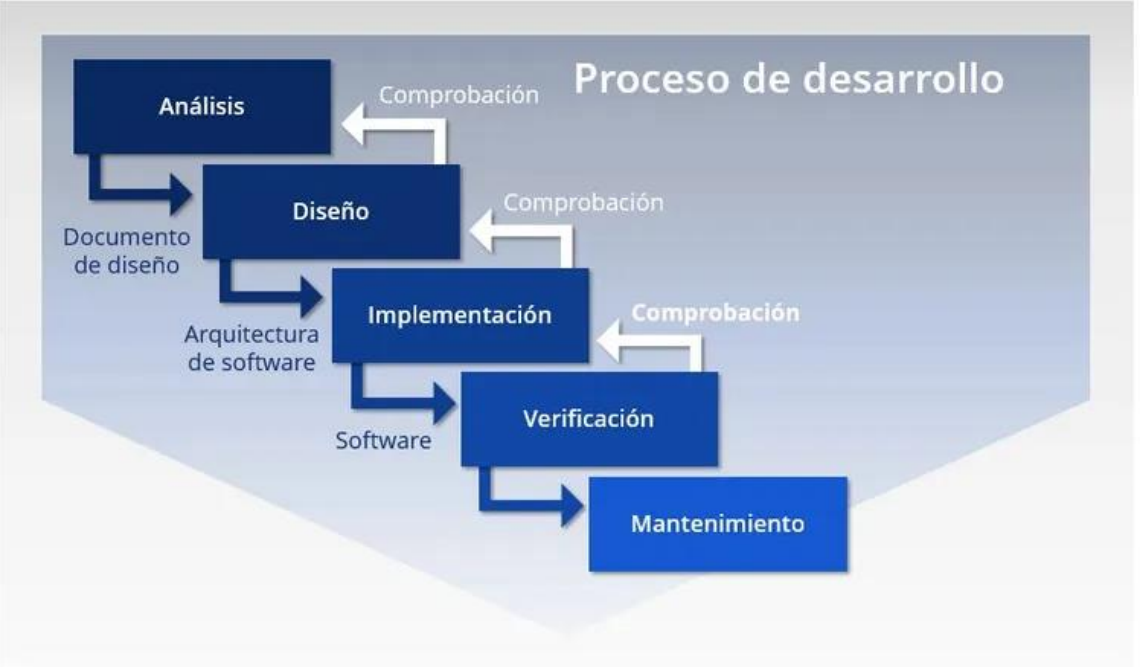
La universidad que forma emprendedores...

Modelos de ciclo de vida de desarrollo de software

Cada modelo de desarrollo de software describe un proceso desde una perspectiva única.

1. El modelo de cascada

El modelo en cascada organiza las fases del desarrollo de manera estricta y ordenada. Su peculiaridad radica en que, hasta que no finaliza una etapa, no comienza la siguiente.



Fases del Modelo Cascada

Fase de análisis: Planificación, análisis y especificación de los requisitos.

Fase de diseño: Diseño y especificación del sistema.

Fase de implementación: Programación y pruebas unitarias.

Fase de verificación: Integración de sistemas, pruebas de sistema e integración.

Fase de mantenimiento: Entrega, mantenimiento y mejora.

Modelos de ciclo de vida de desarrollo de software

Características del Modelo Cascada

- Lineal y secuencial: Una fase debe completarse antes de iniciar la siguiente.
- Fácil de entender y gestionar: Su estructura simple lo hace comprensible para todos los miembros del equipo.
- Requiere una definición clara de requisitos al inicio del proyecto.
- Poco flexible: Los cambios en los requisitos pueden ser costosos y demorados.

Ventajas del Modelo en Cascada

Simple y fácil de entender: Su estructura lineal lo hace sencillo de comprender y gestionar.

Adecuado para proyectos pequeños y con requisitos bien definidos: Funciona bien cuando los requisitos del proyecto son claros y estables desde el inicio.

Promueve una buena documentación: La documentación detallada facilita la comprensión del sistema y el mantenimiento futuro.

Modelos de ciclo de vida de desarrollo de software

Desventajas del Modelo en Cascada

Poco flexible: No se adapta bien a los cambios en los requisitos.

Riesgo de errores: Los errores no se detectan hasta las últimas fases, lo que puede resultar en costos de corrección elevados.

El cliente no ve resultados hasta el final del proyecto: Puede ser difícil para el cliente visualizar el progreso y proporcionar feedback.

No es adecuado para proyectos complejos o con requisitos cambiantes.

¿Cuándo usar el Modelo Cascada?

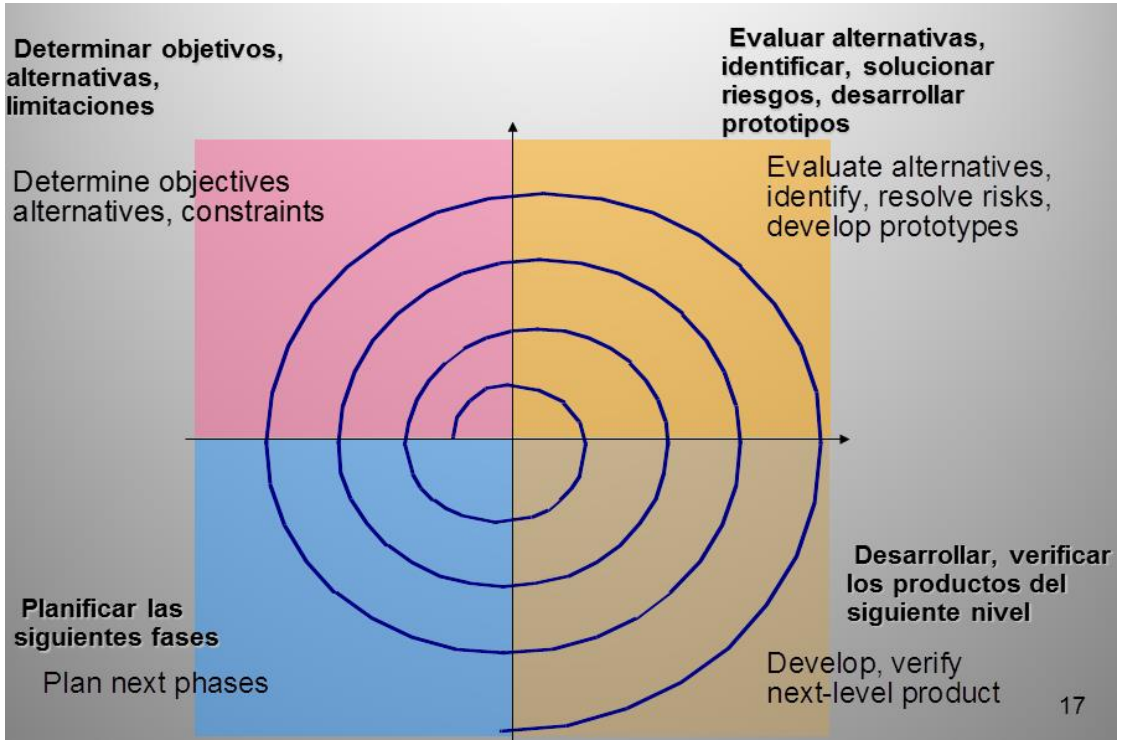
- Cuando tienes una idea clara de cómo quieres que sea el resultado final.
- Cuando los clientes no pueden alterar el alcance de un proyecto una vez que ha comenzado.
- Cuando se trata de éxito, el concepto y la definición son cruciales (pero no la velocidad).
- Cuando no hay dudas sobre lo que hay que hacer.

Modelos de ciclo de vida de desarrollo de software

2. Modelo espiral

Modelo espiral es un tipo de Modelo de desarrollo de software en el que las actividades se crean en espiral y se llevan a cabo en el orden en que se eligen en función del análisis de riesgo.

En cada iteración de este modelo, los objetivos o alternativas deben elegirse en función de las características, que incluyen la experiencia personal, los criterios a satisfacer y las formas de gestión del sistema.



La forma angular, que representa únicamente el desarrollo del software dentro del proyecto, y la forma radial, que indica el crecimiento en costo ya que cada iteración tarda más en terminar.

Modelos de ciclo de vida de desarrollo de software

Fases del Modelo Espiral

Fase de planificación: El paso inicial es identificar y establecer objetivos y metas a alcanzar. Luego, como alternativas, presentan las mejores formas potenciales de satisfacer los objetivos. Todo esto requiere una comunicación continua entre el cliente y el equipo de gestión del proyecto.

Fase de análisis de riesgos: Al planificar y finalizar la estrategia de reducción de riesgos, se identifican los posibles peligros. Cada peligro destacado se somete a un examen exhaustivo. Se pueden crear prototipos para eliminar la posibilidad de requisitos ambiguos. Los riesgos se minimizan tomando precauciones.

Fase de ingeniería: Implica la codificación, prueba e implementación del software. Tras una evaluación de riesgos, se adopta el modelo de desarrollo. El modelo a utilizar está determinado por el nivel de riesgo que se ha reconocido para esa fase.

Fase de evaluación: Valoración del cliente sobre el programa. Se decide si repetir o no el ciclo. Aquí se está planificando la siguiente fase del proyecto.

Modelos de ciclo de vida de desarrollo de software

Características Modelo Espiral

- Gestión de riesgos proactiva: Identifica y mitiga riesgos en cada iteración.
- Flexible: Permite adaptarse a los cambios en los requisitos.
- Requiere una experiencia significativa en la gestión de proyectos.

Ventajas del Modelo en Espiral

Gestión de riesgos: Permite identificar y mitigar los riesgos de manera proactiva.
Flexibilidad: Se adapta bien a los cambios en los requisitos.
Calidad: Permite entregar productos de alta calidad.
Visibilidad del progreso: El cliente puede ver el progreso del proyecto en cada iteración.

Desventajas del Modelo en Espiral

Complejidad: Requiere una planificación y gestión cuidadosas.
Dependencia de la evaluación de riesgos: La efectividad del modelo depende de una evaluación precisa de los riesgos.
Puede ser costoso: Si no se gestiona correctamente, puede llevar a costos elevados.

Modelos de ciclo de vida de desarrollo de software

¿Cuándo usar el Modelo Espiral?

Las ventajas del modelo en espiral son más evidentes en situaciones en las que:

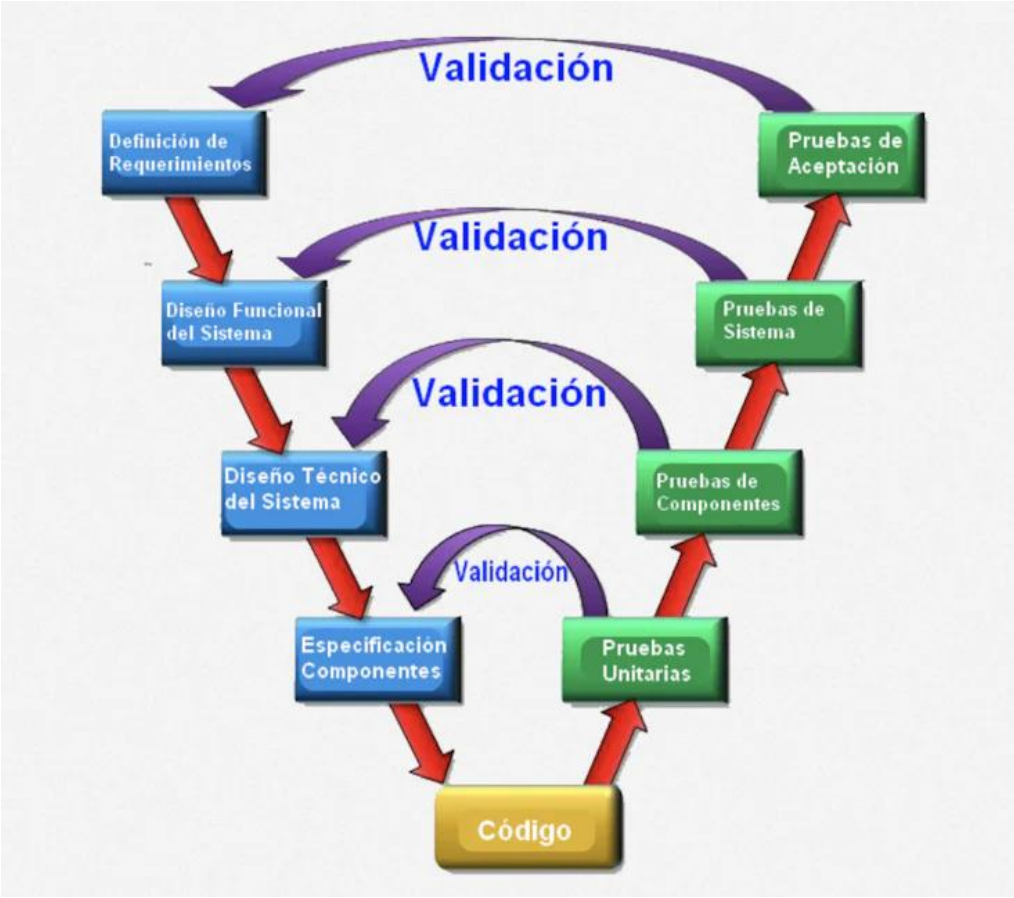
- Es deseable tener lanzamientos de software frecuentes.
- Se utilizan prototipos.
- La gestión de riesgos y gastos es fundamental.
- En proyectos de riesgo medio-alto y riesgo alto.
- Los criterios de requisitos son ambiguos y difíciles de entender.
- Se están produciendo muchos cambios, y pueden ocurrir en cualquier momento.
- Ya sea por razones económicas o de otro tipo, el compromiso del proyecto a largo plazo se ve comprometido.

Modelos de ciclo de vida de desarrollo de software

3. Modelo V

El modelo V, también conocido como el modelo de cuatro niveles, es un concepto utilizado en una variedad de procesos de desarrollo, como el desarrollo de software.

La El modelo V proporciona apoyar los métodos de gestión de la calidad y describe cómo estas distintas etapas pueden interactuar entre sí, además de la fases de desarrollo del proyecto. Recibe su nombre de la forma de su cuerpo, que se asemeja a la letra V.



Modelos de ciclo de vida de desarrollo de software

Fases del Modelo V:

Fase de Verificación:

- **Análisis de requisitos:** El paso inicial de la fase de verificación es comprender las expectativas de los clientes sobre nuestros productos mediante una amplia comunicación con los clientes.
- **Diseño de sistemas:** Después de la identificación de los requisitos de los clientes y las expectativas de nuestros productos, se debe desarrollar el sistema de diseño detallado para el desarrollo del producto.
- **Diseño arquitectónico:** El diseño del sistema se segrega en diferentes módulos según sus funcionalidades. Se reconoce la transferencia de datos entre los módulos internos y otros sistemas.
- **Diseño del módulo:** Los diseños se segregan aún más en módulos más pequeños y más detallados.

Modelos de ciclo de vida de desarrollo de software

Fases del Modelo V:

Fase de Validación:

Examen de la unidad: Las pruebas unitarias eliminan errores a nivel de código o de unidad.

Pruebas de integración: Las pruebas de integración validan la comunicación interna entre módulos dentro del sistema.

Pruebas del sistema: Las pruebas del sistema examinan los requisitos funcionales y no funcionales de la aplicación desarrollada.

Pruebas de aceptación del usuario (UAT): UAT valida la usabilidad del sistema desarrollado en el mundo real.

Modelos de ciclo de vida de desarrollo de software

Características del Modelo V

- Relación entre cada fase de desarrollo y su correspondiente fase de prueba.
- Énfasis en la calidad del producto final.
- Requiere una planificación detallada desde el inicio.

Ventajas del Modelo V

Estructura clara: Facilita la planificación y el seguimiento del proyecto.
Énfasis en las pruebas: Ayuda a garantizar la calidad del producto.
Documentación detallada: Proporciona un registro completo del desarrollo.
Adecuado para proyectos pequeños y medianos con requisitos estables.

Desventajas del Modelo V

Rigidez: Poco flexible ante cambios en los requisitos.Dificultad para manejar proyectos complejos: Puede resultar engorroso para proyectos grandes y con requisitos cambiantes.
Pruebas al final del ciclo: Los errores se detectan tarde, lo que puede aumentar los costos de corrección.

Modelos de ciclo de vida de desarrollo de software

¿Cuándo usar el Modelo V?

El modelo V debe utilizarse en las siguientes circunstancias.

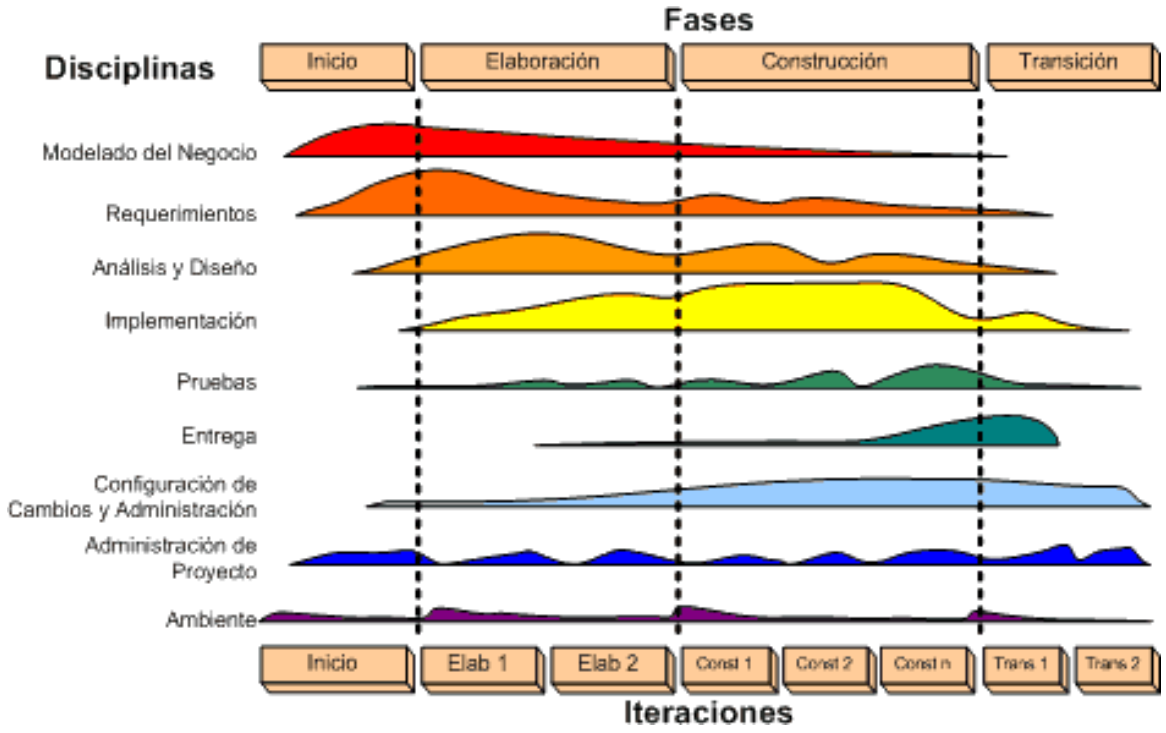
- Cuando los requisitos y objetivos son explícitos e inequívocos.
- Cuando los requisitos técnicos tales como recursos técnicos y expertos técnicos estén disponibles a la mano.
- Cuando las fallas del sistema desarrollado son aceptables.

Modelos de ciclo de vida de desarrollo de software

4. El proceso unificado racional (RUP)

La Proceso Racional Unificado (RUP) es un desarrollo de aplicaciones de software enfoque que incluye una serie de herramientas para ayudar en la codificación del producto final y las actividades que lo acompañan. RUP es una metodología orientada a objetos para gestión de proyectos y desarrollo de software de alta calidad.

El RUP es un conjunto de enfoques ajustables al entorno y exigencias de cada empresa, más que un sistema con procesos rígidos.



Modelos de ciclo de vida de desarrollo de software

Fases del modelo Rational Unified Process (RUP)

Fase de Inicio: Se define el alcance del proyecto, se identifican los stakeholders y se crea un plan de proyecto inicial.
Fase de Elaboración: Se refina la visión del producto, se crea un modelo de arquitectura inicial y se identifican los principales riesgos.
Fase de Construcción: Se desarrolla el producto, se realizan pruebas y se prepara para la entrega.
Fase de Transición: Se entrega el producto al cliente, se proporciona soporte y se realizan las actividades de post-implementación.

Modelos de ciclo de vida de desarrollo de software

Características del modelo Rational Unified Process (RUP)

Iterativo e incremental: Al igual que otros modelos de este tipo, RUP divide el desarrollo en ciclos cortos (iteraciones) y entrega versiones incrementales del producto. Esto permite una mayor flexibilidad y adaptación a los cambios.

Basado en casos de uso: Los casos de uso son la unidad central de RUP y sirven para capturar los requisitos funcionales del sistema desde la perspectiva del usuario.

Centrado en la arquitectura: RUP enfatiza la importancia de una arquitectura sólida desde el inicio del proyecto.

Dirigido por riesgos: Los riesgos se identifican y mitigan de forma proactiva en cada iteración.

Controlado por el proceso: RUP define un conjunto de procesos, artefactos y roles para guiar el desarrollo del software.

Adaptable: RUP es un proceso genérico que puede ser adaptado a las necesidades específicas de cada proyecto.

Modelos de ciclo de vida de desarrollo de software

Ventajas del del modelo Rational Unified Process (RUP)

Proceso bien definido: Proporciona una guía clara para el desarrollo de software.
Enfoque en la calidad: Integra la gestión de la calidad en todo el proceso.
Adaptable: Puede ser adaptado a diferentes tipos de proyectos.
Soporte de herramientas: Existen muchas herramientas que soportan el RUP.

Desventajas del del modelo Rational Unified Process (RUP)

Complejidad: Puede ser complejo de implementar en proyectos pequeños o con equipos pequeños.
Rigidez: Puede ser percibido como demasiado rígido y burocrático.Curva de aprendizaje: Requiere una inversión significativa en tiempo y esfuerzo para aprender y dominar el proceso.

Modelos de ciclo de vida de desarrollo de software

- ¿Cuándo usar el modelo **del modelo Rational Unified Process** ?
- Cuando hay un cambio constante en los requisitos.
- Cuando se dispone de información y datos veraces.
- Cuando necesitas ciertas integraciones a lo largo del proceso de desarrollo.

Modelos de ciclo de vida de desarrollo de software

5. Modelo incremental e iterativo

El desarrollo de software iterativo e incremental es una técnica de desarrollo de software basada en un patrón cíclico de lanzamiento y actualización y un aumento constante en la adición de funciones.

El desarrollo de software iterativo e incremental comienza con la planificación y continúa a través de ciclos de desarrollo iterativos con comentarios continuos de los usuarios y adiciones de funciones incrementales, que culminan en la implementación del software al final de cada ciclo.

Tipos de modelos iterativos e incrementales

Modelo en espiral: Combina aspectos del modelo en cascada y el prototipado. Cada iteración implica una planificación, análisis de riesgos, desarrollo y evaluación.

Modelo de desarrollo rápido de aplicaciones (RAD): Enfoque en el desarrollo rápido de aplicaciones utilizando herramientas y componentes reutilizables. Cada iteración se centra en un conjunto específico de funcionalidades.

Modelo ágil: Existen diversas metodologías ágiles (Scrum, Kanban, XP) que comparten características como equipos autoorganizados, entregas frecuentes, y adaptación a los cambios.

Modelos de ciclo de vida de desarrollo de software

Fases del Modelo Incremental e Iterativo

Los siguientes pasos se pueden utilizar para clasificar el desarrollo iterativo e incremental:

Fase de Iniciación: La fase de iniciación de un proyecto se ocupa del alcance, las necesidades y los peligros a un nivel superior.
Fase de Elaboración: Crea una arquitectura viable que mitiga los riesgos identificados en la primera fase y cumple con los criterios no funcionales.
Fase de construcción: Gradualmente completa los componentes de la arquitectura con código listo para la producción, que se desarrolla mediante el análisis, la implementación, el diseño y las pruebas de los requisitos funcionales.
Fase de transición: Entregar el sistema al entorno operativo de producción durante la fase de transición.

Modelos de ciclo de vida de desarrollo de software

Características del Modelo Incremental e Iterativo

- Desarrollo en iteraciones: El proyecto se divide en ciclos cortos (iteraciones) de duración fija.
- Entregas incrementales: Al final de cada iteración se entrega una versión funcional del producto con nuevas funcionalidades o mejoras.
- Retrospectiva: Al finalizar cada iteración se realiza una revisión para identificar mejoras y ajustar el plan para la siguiente iteración.
- Adaptación a cambios: Permite incorporar cambios en los requisitos a lo largo del desarrollo.
- Mayor visibilidad: El cliente puede ver el progreso del proyecto y proporcionar feedback de forma temprana.
- Reducción de riesgos: Los riesgos se identifican y mitigan de forma temprana en el proyecto.

Modelos de ciclo de vida de desarrollo de software

Ventajas de los modelos iterativos e incrementales

Mayor flexibilidad: Permite adaptarse a cambios en los requisitos.

Reducción de riesgos: Los riesgos se identifican y mitigan de forma temprana.

Mayor satisfacción del cliente: El cliente puede ver el progreso del proyecto y proporcionar feedback de forma temprana.

Mejor calidad: Al identificar y corregir errores en cada iteración, se mejora la calidad del producto final.

Mayor productividad: Los equipos pueden concentrarse en un conjunto limitado de funcionalidades en cada iteración.

Desventajas de los modelos iterativos e incrementales

Requiere una planificación cuidadosa: Cada iteración debe planificarse y ejecutarse de manera eficiente.

Mayor necesidad de comunicación: La comunicación entre el equipo de desarrollo y el cliente es crucial.Puede ser más difícil de gestionar para proyectos muy grandes y complejos.

Modelos de ciclo de vida de desarrollo de software

Cuándo utilizar modelos iterativos e incrementales

Proyectos con requisitos cambiantes: Cuando los requisitos no están completamente definidos al inicio del proyecto.
Proyectos grandes y complejos: Al dividir el proyecto en iteraciones más pequeñas, es más fácil de gestionar.
Proyectos con alta prioridad en la entrega temprana de valor: Permite entregar funcionalidades al cliente de forma temprana.

Modelos de ciclo de vida de desarrollo de software

6. Modelo Prototipo

Al crear un software o aplicación, es típico utilizar un modelo prototipo para ofrecer una versión anterior y funcional que pueda utilizarse como presentación o muestra del proyecto.

La creación de prototipos es una excelente manera de recibir información sobre los requisitos, la funcionalidad y la operabilidad, de modo que el desarrollo final del producto pueda avanzar de manera más rápida y eficiente.

A **modelo prototipo** es una aplicación funcional del producto que da una idea de las características fundamentales del producto o sistema final.

Modelos de ciclo de vida de desarrollo de software

Fases del Modelo Prototipo

Fase de Análisis de Requisitos: El paso inicial del modelo trata de establecer los requisitos del sistema deseable.

Fase de Diseño: Después de la identificación de los requisitos del sistema deseado, se forma un diseño conceptual básico.

Fase de Construcción de Prototipos: Con la ayuda del diseño conceptual básico, se construye un prototipo de trabajo para el sistema deseado.

Fase de Evaluación Inicial: El prototipo es probado por el cliente en este paso para evaluar funcionalidades y limitaciones.

Fase de Prototipo de Refinación: El prototipo se refina aún más, analizando la evaluación realizada por el cliente.

Fase Producción: Una vez que se ejecuta el proceso de refinación, se produce el sistema final para su uso en tiempo real.

Modelos de ciclo de vida de desarrollo de software

Ventajas del Modelo Prototipo

Reducción de riesgos: Al obtener feedback temprano, se pueden identificar y corregir problemas en una etapa temprana del desarrollo.
Mejora de la comunicación: El prototipo facilita la comunicación entre el equipo de desarrollo y los usuarios.
Mayor satisfacción del cliente: Los usuarios se sienten más involucrados en el proceso de desarrollo.
Flexibilidad: Permite realizar cambios en los requisitos de manera más fácil.

Modelos de ciclo de vida de desarrollo de software

Ventajas del Modelo Prototipo

Puede llevar a un desarrollo superficial: Si no se gestiona correctamente, puede llevar a un producto final con una arquitectura débil.
Puede generar expectativas poco realistas: Un prototipo puede dar una impresión errónea sobre las capacidades del producto final.
Requiere una buena comunicación: Es fundamental una buena comunicación entre el equipo de desarrollo y los usuarios.

Cuando utilizar el modelo prototipo

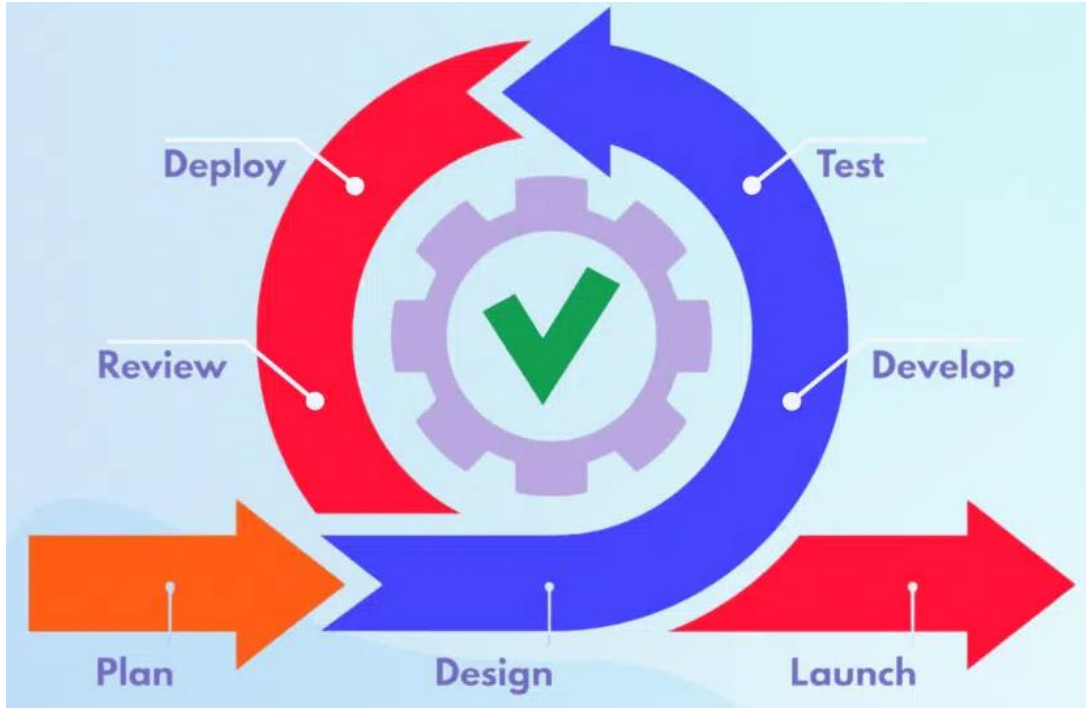
Proyectos con requisitos inciertos: Cuando los requisitos no están completamente definidos al inicio.
Interfaces de usuario: Para diseñar interfaces de usuario intuitivas y fáciles de usar.
Validación de conceptos: Para verificar la viabilidad de una idea antes de invertir mucho tiempo y recursos en su desarrollo.

Modelos de ciclo de vida de desarrollo de software

Enfoque Ágil

Las empresas que se dedican a una transformación digital completa terminan aplicando y desarrollando enfoques ágiles dentro de sus departamentos para ofrecer bienes y/o servicios de mayor calidad a menores costos y en menos tiempo.

El enfoque ágil para el desarrollo de software tiene como objetivo proporcionar sistemas de software que funcionen en un corto período de tiempo.



Desarrollo Ágil de Software Los enfoques, en particular, tienen como objetivo ofrecer pequeños fragmentos de software en funcionamiento en un corto período de tiempo para mejorar la satisfacción del cliente. Para lograr un desarrollo continuo, estas estrategias emplean enfoques flexibles y cooperación.

Modelos de ciclo de vida de desarrollo de software

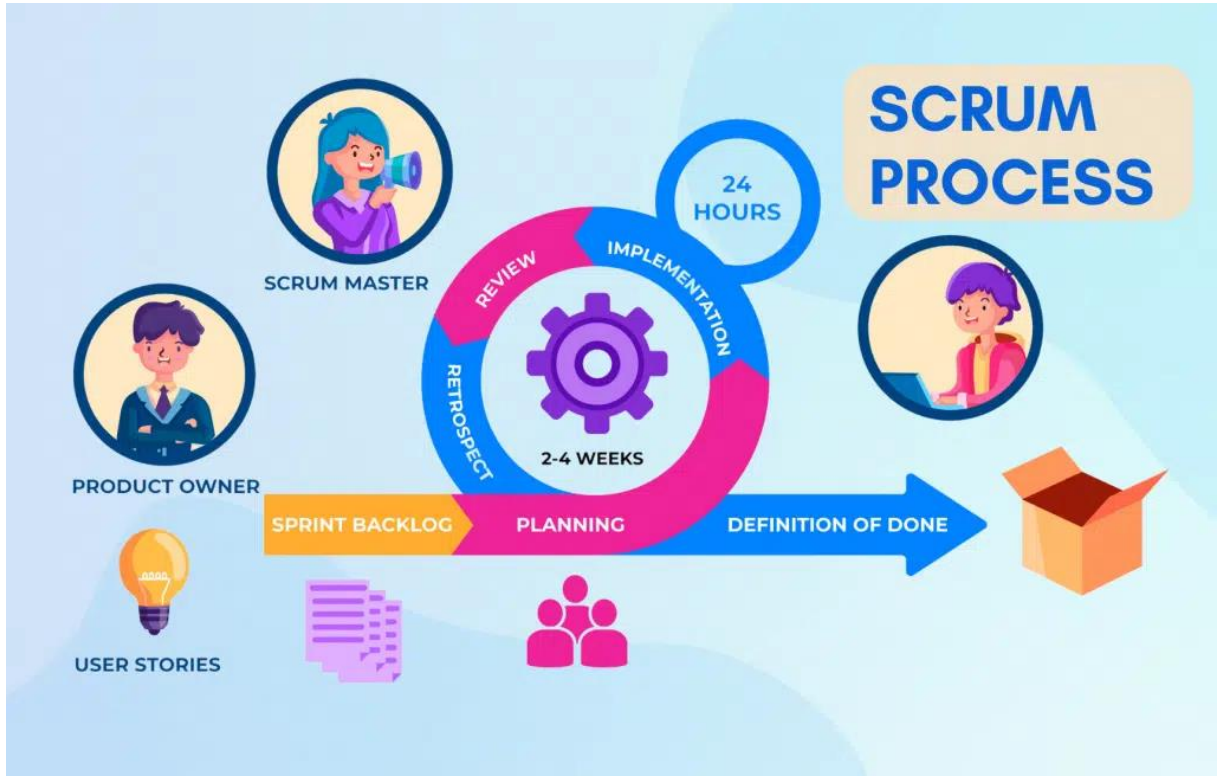
Las metodologías ágiles se pueden clasificar en diferentes tipos, tales como:

7. Modelo SCRUM

Al abordar los desafíos, los proyectos que utilizan esta técnica otorgan un gran valor al intelecto, la experiencia y las habilidades que los miembros del equipo de desarrollo aportan.

Las actividades del proyecto se completan en ciclos cortos conocidos como sprints, que son relativamente manejables y bien priorizados, lo que permite un fácil seguimiento del progreso.

Comparado con otros modelos de desarrollo de software, esta estrategia beneficiaría a iniciativas más grandes y una de las razones es que los desarrolladores se sienten dedicados a los objetivos y responsables del éxito de la iniciativa.



Modelos de ciclo de vida de desarrollo de software

Fases del Modelo Scrum

1. Planificación:

- **Creación del Product Backlog:** Se crea una lista priorizada de todas las características y funcionalidades que el producto final debe tener.
- **Planificación del Sprint:** El equipo selecciona un conjunto de elementos del Product Backlog para trabajar durante el próximo Sprint.

2. Desarrollo:

- **Daily Scrum:** Una reunión diaria de 15 minutos donde el equipo sincroniza el trabajo y planifica las próximas 24 horas.
- **Desarrollo del Incremento:** El equipo trabaja en las tareas del Sprint Backlog para crear un incremento del producto.

3. Revisión:

- **Revisión del Sprint:** Al final del Sprint, el equipo demuestra el Incremento al Product Owner y a las partes interesadas.

Modelos de ciclo de vida de desarrollo de software

Fases del Modelo Scrum

4. Retrospectiva:

- **Retrospectiva del Sprint:** El equipo reflexiona sobre lo que fue bien, lo que no funcionó y qué se puede mejorar en el próximo Sprint.

5. Refinamiento del Product Backlog:

- **Refinamiento del Product Backlog:** El Product Backlog se actualiza continuamente para reflejar los nuevos requisitos y la retroalimentación del cliente.

Características de Scrum

- Iterativo e incremental: El desarrollo se divide en ciclos cortos llamados "sprints" (generalmente de 2 a 4 semanas), al final de cada uno se entrega un producto funcional incrementado.
- Autoorganizado: Los equipos de desarrollo son autoorganizados y toman decisiones sobre cómo abordar el trabajo.
- Colaborativo: Fomenta la colaboración entre los miembros del equipo y con el cliente.
- Adaptable: Se adapta a los cambios en los requisitos a lo largo del proyecto.
- Enfoque en el valor: Se centra en entregar valor al cliente lo antes posible.

Modelos de ciclo de vida de desarrollo de software

Ventajas de Scrum

Mayor flexibilidad: Se adapta bien a los cambios en los requisitos.

Entrega temprana y frecuente de valor: Los clientes pueden ver resultados más rápidamente.

Mayor satisfacción del cliente: Al involucrar al cliente en el proceso, se asegura que el producto final cumpla con sus expectativas.

Mejor calidad del producto: El enfoque en la mejora continua conduce a productos de mayor calidad.

Mayor compromiso del equipo: Los equipos se sienten más motivados y comprometidos con el proyecto.

Modelos de ciclo de vida de desarrollo de software

Desventajas de Scrum

Requiere un cambio cultural: Puede ser difícil adaptarse a Scrum si el equipo está acostumbrado a trabajar de forma tradicional.

Necesidad de un Product Owner comprometido: El Product Owner debe estar disponible y comprometido con el proyecto.

Puede ser difícil estimar el trabajo al principio: Puede ser complicado estimar la cantidad de trabajo que se puede realizar en un Sprint al inicio del proyecto.

No es adecuado para todos los proyectos: Scrum funciona mejor en proyectos complejos y cambiantes, pero puede no ser la mejor opción para proyectos muy pequeños o muy estables.

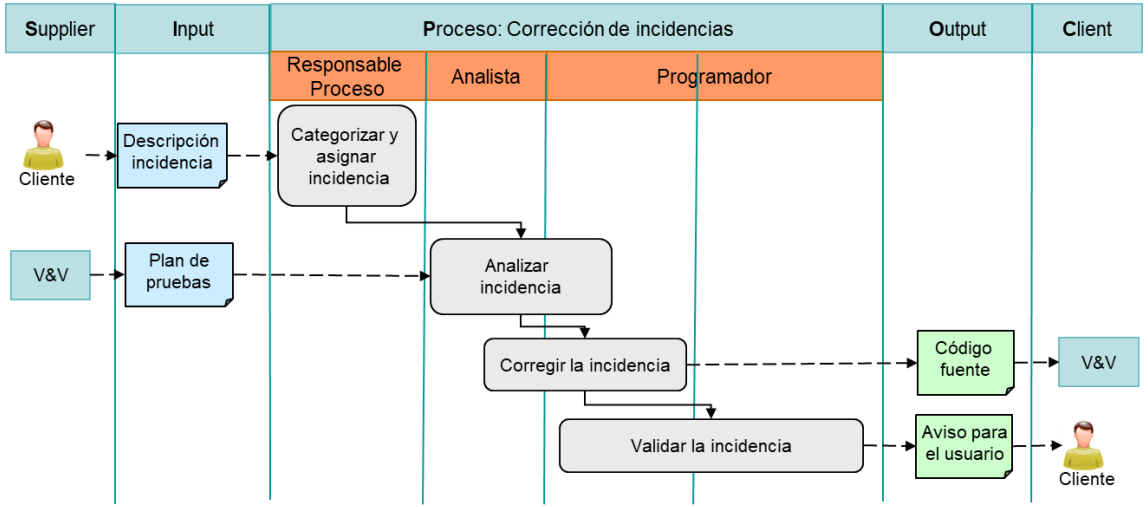
¿Cuándo usar Scrum Agile Model?

- Este enfoque se utiliza en situaciones donde se requieren resultados inmediatos.
- En los casos en que hay mucha ambigüedad y los deberes no están bien definidos.
- Cuando un cliente solicita un enfoque de desarrollo altamente personalizado para un determinado producto.

Modelos de ciclo de vida de desarrollo de software

8. Modelo Kanban

Kanban es un marco bien conocido para ágil y Desarrollo de software DevOps. Requiere comunicación de capacidad en tiempo real y apertura de trabajo completa.
Kanban es un enfoque flexible de la gestión del trabajo visual que cambia a medida que cambian las necesidades del equipo.



Kanban ayuda en la visualización del trabajo para que pueda entenderse mejor, mostrarse a otros y mantener actualizados a los interesados. Como resultado, podemos asegurar que el servicio es capaz de hacer la tarea que requiere el cliente.

Modelos de ciclo de vida de desarrollo de software

Fases del Modelo Kanban (más bien, etapas del flujo de trabajo)

- Tareas por hacer (To Do): Las tareas se añaden a esta columna cuando se crean.
- En progreso (In Progress): Las tareas se mueven a esta columna cuando se están trabajando.
- Revisar (Review): Las tareas se revisan antes de ser marcadas como completadas.
- Hecho (Done): Las tareas se marcan como completadas cuando cumplen con los criterios de aceptación.

Características del Modelo Kanban

- Visualización: Utiliza un tablero Kanban para visualizar el flujo de trabajo, desde las tareas pendientes hasta las completadas.
- Flujo continuo: No hay divisiones rígidas en iteraciones como en Scrum, el trabajo fluye de manera continua.
- Limitar el trabajo en proceso (WIP): Se establece un límite en la cantidad de tareas que pueden estar en progreso en cada etapa, lo que evita sobrecargar al equipo.
- Mejora continua: Se realizan ajustes al proceso de forma incremental para optimizar el flujo de trabajo.
- Foco en la entrega: Se prioriza la entrega de valor al cliente de manera rápida y eficiente.

Modelos de ciclo de vida de desarrollo de software

Ventajas del Modelo Kanban

Sencillo de implementar: Puede adaptarse a cualquier proceso existente, sin necesidad de grandes cambios.

Flexible: Permite ajustar el flujo de trabajo según las necesidades del equipo y del proyecto.

Visualización clara del trabajo: El tablero Kanban proporciona una visión clara del estado de cada tarea y del flujo de trabajo en general.

Mejora continua: Fomenta la identificación y resolución de cuellos de botella en el proceso.

Menos burocrático: No tiene las ceremonias formales de Scrum, lo que puede ser más atractivo para algunos equipos.

Desventajas del Modelo Kanban

Falta de estructura: Puede ser menos estructurado que Scrum, lo que puede dificultar la gestión de proyectos grandes o complejos.

Requiere disciplina: Los equipos deben ser disciplinados para mantener actualizado el tablero y seguir las reglas del sistema.

Puede ser difícil medir el progreso: Sin los Sprints de Scrum, puede ser más difícil medir el progreso del proyecto.

Modelos de ciclo de vida de desarrollo de software

Cuándo usar el modelo ágil de Kanban

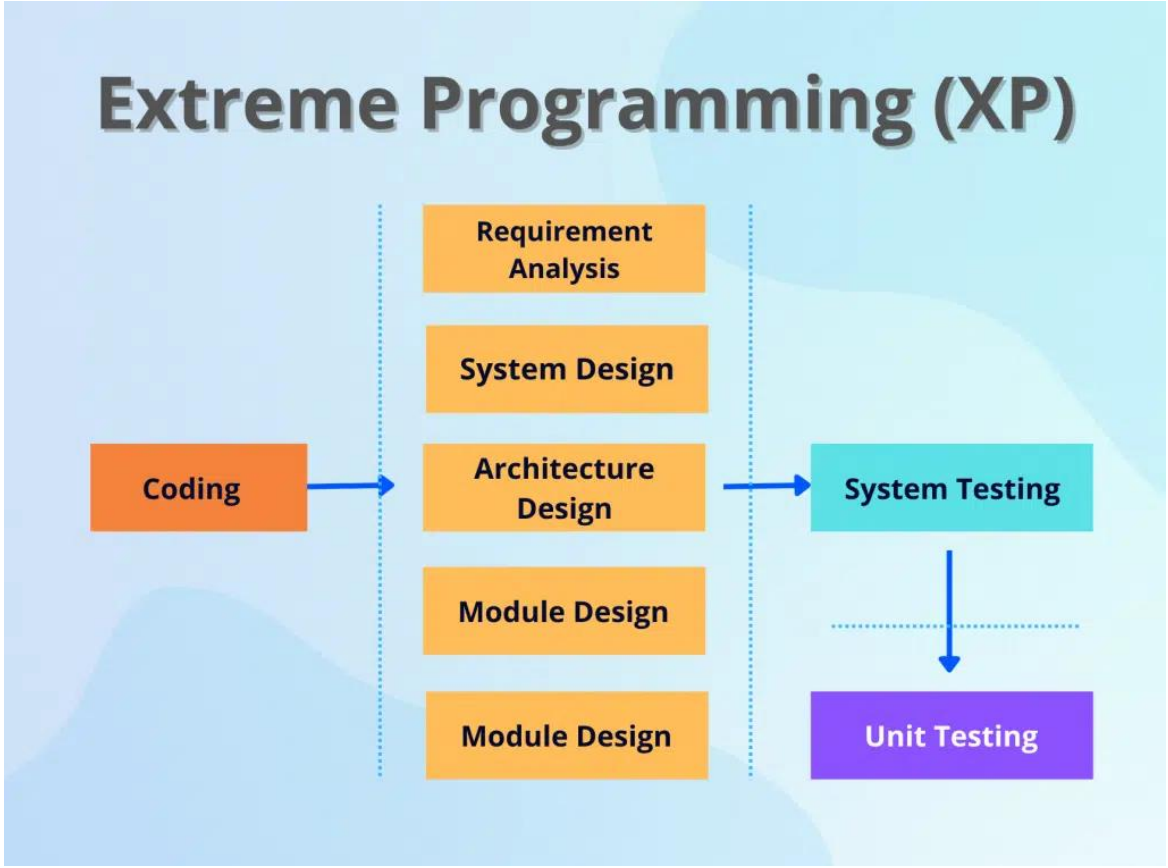
- Cuando necesite eliminar procesos y prácticas innecesarias.
- Cuando necesite un modelo que proporcione un flujo fluido del proceso de desarrollo.
- Cuando se busca la mejora continua del sistema.

Modelos de ciclo de vida de desarrollo de software

9. Programación extrema (XP)

La técnica de programación extrema permite a los especialistas realizar cambios incluso después de que haya comenzado la iteración. Normalmente toma de 1 a 2 semanas completar una iteración.

El XP o Enfoque de programación extrema es una metodología de desarrollo ágil con el objetivo de desarrollar y gestionar proyectos con eficiencia, flexibilidad y control. Se basa en la comunicación, la reutilización del código generado y la retroalimentación.



Modelos de ciclo de vida de desarrollo de software

Fases del Modelo de Programación Extrema (XP):

- Planificación: Las historias de usuarios se priorizan y se dividen en miniversiones según su identidad. Habrá una reevaluación de la planificación.
- Codificación: Trabajar con un código simple en esta fase, realizando solo el mínimo absoluto para que funcione. Será posible conseguir el prototipo.
- Pruebas: La programación se realiza en parejas frente a la misma computadora, “a dos manos”. Es común que los socios se cambien. Esto asegura que se crea un código más general, que cualquier otro programador puede comprender y trabajar con él.
- Lanzamiento: Si hemos llegado a esta fase, indica que hemos probado con éxito todas las historias de usuario o versiones mini considerando las necesidades del cliente.

Modelos de ciclo de vida de desarrollo de software

Características del Modelo XP

Iterativo e incremental: El desarrollo se divide en ciclos cortos, al igual que en Scrum, para permitir una rápida adaptación a los cambios.

Simpleza: Se busca la solución más sencilla para cada problema, evitando la complejidad innecesaria.

Comunicación constante: Se fomenta la comunicación abierta y frecuente entre todos los miembros del equipo y el cliente.

Retroalimentación continua: Se obtiene feedback constante del cliente para asegurar que el software se está desarrollando según sus necesidades.

Mejora continua: Se busca mejorar continuamente el proceso de desarrollo a través de la reflexión y la adaptación.

Modelos de ciclo de vida de desarrollo de software

Ventajas del Modelo XP

Adaptabilidad a los cambios: Al ser iterativo e incremental, XP permite adaptarse fácilmente a los cambios de requisitos.

Alta calidad del software: Las prácticas de XP, como las pruebas y la refactorización, contribuyen a una mayor calidad del software.

Mayor satisfacción del cliente: La participación activa del cliente y la entrega frecuente de funcionalidades incrementan la satisfacción.

Mejor colaboración en equipo: El trabajo en parejas y la comunicación constante fomentan la colaboración.

Mayor productividad: Al eliminar el exceso de documentación y burocracia, XP puede aumentar la productividad.

Modelos de ciclo de vida de desarrollo de software

Desventajas del Modelo XP

Requiere un alto nivel de compromiso: Todos los miembros del equipo deben estar comprometidos con las prácticas de XP.

Puede no ser adecuado para todos los proyectos: XP funciona mejor en proyectos pequeños y medianos con requisitos cambiantes.

Puede ser difícil de implementar en organizaciones grandes: La cultura organizacional puede ser un obstáculo para la adopción de XP.

Requiere una buena comunicación: La comunicación efectiva es esencial para el éxito de XP.

Modelos de ciclo de vida de desarrollo de software

¿Cuándo usar el Modelo de Programación Extrema (XP)?

Este enfoque se puede utilizar cuando se requieren los siguientes factores:

- La comunicación entre el cliente y el equipo de desarrollo es siempre abierta.
- El cambio constante requiere una reacción rápida.
- Con un calendario flexible de actividades, la planificación está abierta.
- El software funcional tiene prioridad sobre todas las demás formas de documentación.
- Los principales criterios de éxito del proyecto son las necesidades del cliente y los esfuerzos del equipo del proyecto.
- Colabore de forma remota en proyectos.