
Metodologías
y Procesos
de Análisis
de Software

2. Metodologías y procesos de análisis de software

La evolución de la disciplina de ingeniería de software ha traído consigo diferentes propuestas para mejorar los resultados en la búsqueda de la metodología adecuada para producir software de calidad en cualquier contexto de desarrollo. Las metodologías se caracterizan por exponer procesos basados en planeación exhaustiva. Esta planeación se realiza esperando que el resultado de cada proceso sea determinante y predecible. La experiencia ha mostrado que, como consecuencia de las características del software, los resultados de los procesos no son siempre predecibles y sobre todo, es difícil prever desde el comienzo del proyecto cada resultado.

El modelo de cascada fue uno de los primeros modelos de ciclo de vida que formalizó un conjunto de procesos de desarrollo de software. Este modelo describe un orden secuencial en la ejecución de los procesos asociados. El modelo espiral se postuló como una alternativa al modelo de cascada, la ventaja de este modelo radica en el perfeccionamiento de las soluciones encontradas con cada ciclo de desarrollo, en términos de dar respuesta a los requerimientos inicialmente analizados. El modelo de cascada y el modelo espiral suponen, de manera general, que los requerimientos del cliente no cambian radicalmente en el transcurso del desarrollo del sistema.

Así mismo, la realización de prototipos es una herramienta en la que se apoyan diferentes metodologías. Un prototipo debe tener el objetivo de mostrar al cliente o a la gerencia del proyecto el resultado que se obtendrá de la implementación de cada uno de los requerimientos del cliente una vez terminado el desarrollo. Con los prototipos se tiene la posibilidad de obtener retroalimentación de manera oportuna.

La solución a algunos de los problemas presentados por las metodologías tradicionales se logra con una gran evolución del modelo espiral. El proceso unificado propone la elaboración de varios ciclos de desarrollo, donde cada uno finaliza con la entrega al cliente de un producto terminado. Éste se enmarca entre los conocidos modelos iterativo-incremental.

Una alternativa para el desarrollo de sistemas, es el modelo XP el cual se caracteriza por ser dirigido por la generación de código y prototipos. Estos modelos para el desarrollo de sistemas son los más representativos, los cuales son: en cascada, espiral, prototipo, incremental, XP y RUP (capitulo V).

2.1 Procesos de desarrollo de la metodología

En un proyecto de desarrollo de software la metodología (Figura 2.1), define "quién debe hacer qué, cuándo y cómo debe hacerlo", no existe una metodología de software universal. Las características de cada proyecto (equipo de desarrollo, recursos, etc.) exigen que el proceso sea configurable.

Figura 2.1 Proceso de desarrollo de software

2.1.1 Método en cascada

Es el enfoque metodológico que ordena rigurosamente las etapas del ciclo de vida del software, de forma tal que el inicio de cada etapa debe esperar a la finalización de la inmediatamente anterior (Figura 2.2).

De esta forma, cualquier error de diseño detectado en la etapa de prueba conduce necesariamente al rediseño y nueva programación del código afectado, aumentando los costos del desarrollo.

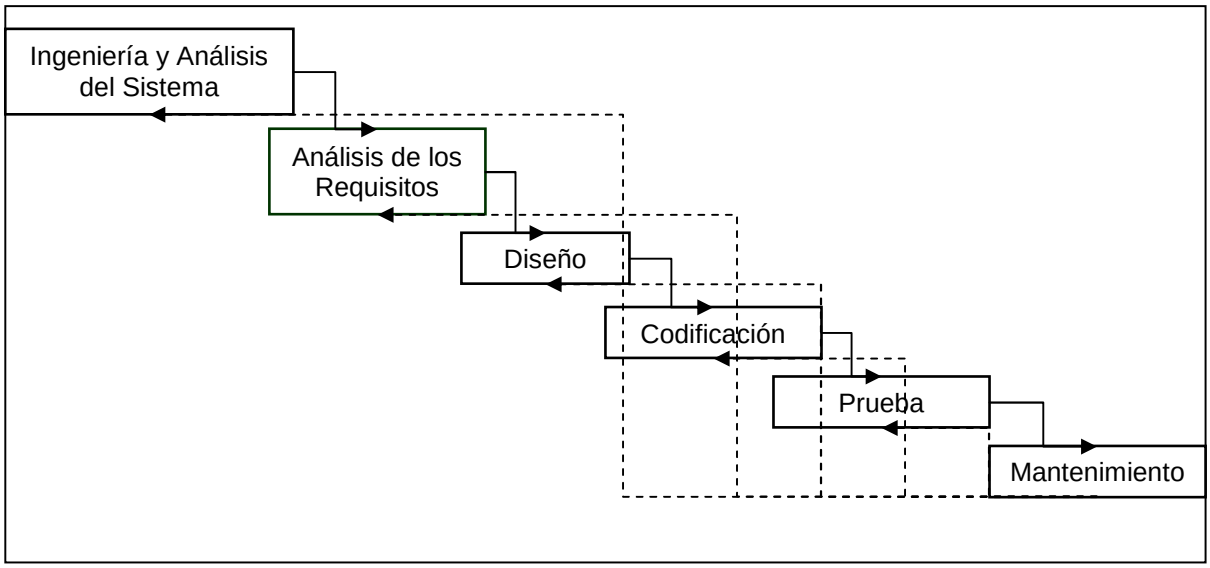

Figura 2.2. Modelo de en cascada

- ***Fase de ingeniería y análisis del sistema.*** Debido a que el software es siempre parte de un sistema mayor el trabajo comienza estableciendo los requisitos de todos los elementos del sistema y luego asignando algún subconjunto de estos requisitos al software.

- ***Fase de análisis de los requisitos.*** Se analizan las necesidades de los usuarios finales del software a desarrollar para determinar qué objetivos debe cubrir. De esta fase surge una memoria llamada SRD (Documento de Especificación de Requisitos), que contiene la especificación completa de lo que debe hacer el sistema sin entrar en detalles internos. Es importante señalar que en esta etapa se deben verificar todo lo que se requiere en el sistema y será aquello lo que seguirá en las siguientes etapas, ya que no se pueden solicitar nuevos requisitos a mitad del proceso de elaboración del software

- ***Fase de diseño:*** Se descompone y organiza el sistema en elementos que puedan elaborarse por separado, aprovechando las ventajas del desarrollo en equipo. Como resultado surge el SDD (Documento de Diseño del Software), que contiene la descripción de la estructura global del sistema y la especificación de lo que debe hacer cada una de sus partes, así como la manera en que se combinan unas con otras. Se realizan los algoritmos

necesarios para el cumplimiento de los requerimientos del usuario así como también los análisis necesarios para saber que herramientas usar en la etapa de Codificación.

- ***Fase de codificación.*** Es la fase de programación propiamente dicha. Aquí se desarrolla el código fuente, haciendo uso de prototipos así como pruebas y ensayos para corregir errores. Dependiendo del lenguaje de programación y su versión, se crean las librerías y componentes reutilizables dentro del mismo proyecto para hacer que la programación sea un proceso mucho más rápido.

- ***Fase de pruebas.*** Los elementos, ya programados, se ensamblan para componer el sistema y se comprueba que funciona correctamente antes de ser puesto en explotación. Una vez que se ha generado el código comienza la prueba del programa. La prueba se centra en la lógica interna del software, y en las funciones externas, realizando pruebas que aseguren que la entrada definida produce los resultados que realmente se requieren.

- ***Fase de mantenimiento.*** El software obtenido se pone en producción. Es una de las fases finales del proyecto. En el desarrollo surgen cambios, para corregir errores o bien para introducir mejoras. El software sufre cambios después de que se entrega al cliente. Los cambios ocurrirán debidos a que hayan encontrado errores, a que el software deba adaptarse a cambios del entorno externo (sistema operativo o dispositivos periféricos), o debido a que el cliente requiera ampliaciones funcionales o del rendimiento.

2.1.2 Método de prototipos evolutivos

Se basa en la creación de una implementación parcial de un sistema, para el propósito explícito de aprender sobre los requerimientos del sistema. Un prototipo es construido de una manera rápida tal como sea posible. Esto es dado a los usuarios, clientes o representantes de ellos, posibilitando que ellos experimenten con el prototipo. Estos individuos luego proveen la retroalimentación sobre lo que a ellos les gustó y no les gustó acerca del prototipo proporcionado, quienes capturan en la

documentación actual de la especificación de requerimientos la información entregada por los usuarios para el desarrollo del sistema real (Figura 2.3).

Las fases que comprende el método de prototipos evolutivos son:

- **Fase de Investigación preliminar**. Las metas principales de esta fase son: determinar el problema y su ámbito, la importancia y los efectos sobre la organización por una parte y, por otro lado, identificar una idea general de la solución para realizar un estudio de factibilidad que determine la viabilidad de una solución software.

Esta primera fase del ciclo se divide en tres actividades:

a) Clarificación de requerimientos
b) Estudio de factibilidad
c) Aprobación del requerimiento

Clasificación de requerimientos. El analista debe de observar en forma objetiva lo que ocurre en la empresa, ya que muchas veces los requerimientos no están claramente establecidos, por lo que, el proyecto requerido debe examinarse para determinar precisamente lo que desea la empresa o usuario.
En muchos casos, los usuarios y los analistas de sistemas trabajan conjuntamente, ya que el usuario tiene ideas definidas acerca de la salida requerida del sistema, las entradas necesarias y, posiblemente una noción general de los controles necesarios.

Estudio de factibilidad. Es determinar si el proyecto es viable. Los aspectos para determinar la factibilidad del proyecto son:

a) Factibilidad técnica: Se debe de investigar si se puede realizar el trabajo para el proyecto con el equipo actual, el personal y el software disponible.
b) Factibilidad económica: Se debe de analizar los beneficios y costos que se tendrán con la creación del sistema
c) Factibilidad operativa: Se debe de investigar si el sistema que se desarrolla se pondrá en marcha, si habrá resistencia al cambio por parte de los usuarios.

Aprobación del requerimiento: En muchas empresas tienen varios proyectos que se encuentran en marcha, por lo que la gerencia debe de decidir qué proyectos son más importantes. Posteriormente, cuando se terminan dicha elección de proyectos, puede iniciarse el desarrollo de la aplicación propuesta.

- **Fase de definición de los requerimientos del sistema.** El objetivo de esta etapa es registrar todos los requerimientos y deseos que los usuarios tienen en relación al proyecto bajo desarrollo. Esta etapa es la más importante de todo el modelo, es aquí donde el desarrollador determina los requisitos mediante la construcción, demostración y retroalimentaciones del prototipo.

El objetivo del análisis de sistemas es comprender situaciones, no resolver problemas. Por tanto, se debe conocer el modo de operación del sistema e identificar los requerimientos que tienen los usuarios para modificarlo o proponer uno nuevo análisis. Un requerimiento son características que debe incluirse en el nuevo sistema.

La definición de requerimientos consiste de cinco etapas:

Análisis grueso y especificación. El propósito de esta subfase es desarrollar un diseño básico para el prototipo inicial.

***Diseño y construcción*.** El objetivo de esta subfase es obtener un prototipo inicial. El desarrollador debe concentrarse en construir un sistema con la máxima funcionalidad, poniendo énfasis en la interfaz el usuario.

Evaluación. Esta etapa tiene dos propósitos: extraer a los usuarios la especificación de los requerimientos adicionales del sistema y verificar que el prototipo desarrollado haya cumplido con la definición de requerimientos del sistema. Si los usuarios identifican fallas en el prototipo, entonces el desarrollador simplemente corrige el prototipo antes de la siguiente evaluación. El prototipo es repetidamente modificado y evaluado hasta que todos los requerimientos del sistema hayan sido satisfechos. El proceso de evaluación puede ser dividido en cuatro pasos separados: preparación, demostración, uso del prototipo y discusión de comentarios. En esta fase se decide si el prototipo es aceptado o modificado.

Modificación. Esto ocurre cuando la definición de requerimientos del sistema es alterada en la subfase de evaluación. El desarrollador entonces debe modificar el prototipo de acuerdo a los comentarios hechos por los usuarios.

***Término*.** Una vez que se ha desarrollado un prototipo estable y completo, es necesario ponerse de acuerdo en relación a aspectos de calidad y de representación del sistema.

- **Fase de diseño técnico**. Durante la construcción del prototipo, el desarrollador no ha realizado el diseño detallado. El sistema debe ser entonces rediseñado y documentado según los estándares de la organización y para ayudar al mantenimiento del sistema. Esta fase de diseño técnico tiene dos etapas: por un lado, la producción de una documentación de diseño que especifica y describe la estructura del software, el control de flujo, las interfaces de usuario y las funciones y, como segunda etapa, la producción de todo lo requerido para realizar cualquier mantenimiento futuro al software.

- **Fase de desarrollo y pruebas**. Los cambios son identificados en el diseño técnico, son implementados y probados para asegurar la corrección y completitud de los mismos con respecto a los requerimientos.

- **Fase de Operación y mantenimiento**. Consiste en la instalación del sistema en ambiente del usuario, en este caso, resulta de menor complejidad, ya que se supone que los usuarios han trabajado con el sistema al hacer las pruebas de prototipos. Considera al mantenimiento una fase de menor importancia, ya que a lo largo del desarrollo del proyecto se corrigieron los errores del prototipo, por lo cual el mantenimiento correctivo del software se reduce. Si eventualmente se requiere dar mantenimiento entonces el proceso de prototipado es repetido definiendo un nuevo conjunto de requerimientos.

Figura 2.3 Modelo de desarrollo por prototipos evolutivos

2.1.3 Método incremental

En esta metodología el desarrollo y entrega del sistema se divide en incrementos, con cada incremento se entrega parte de la funcionalidad requerida en el sistema. Los Requerimientos de usuarios son priorizados y la prioridad más alta es incluida en los primeros incrementos (Figura 2.4).

Una vez comenzado el desarrollo de un incremento, los requerimientos son congelados de modo que requerimientos para incrementos posteriores puedan continuar evolucionando.

Se evitan proyectos largos y se entrega "*Algo de valor*" a los usuarios con cierta frecuencia, el usuario se involucra más en el proyecto se evalúan y el resultado puede ser muy positivo.

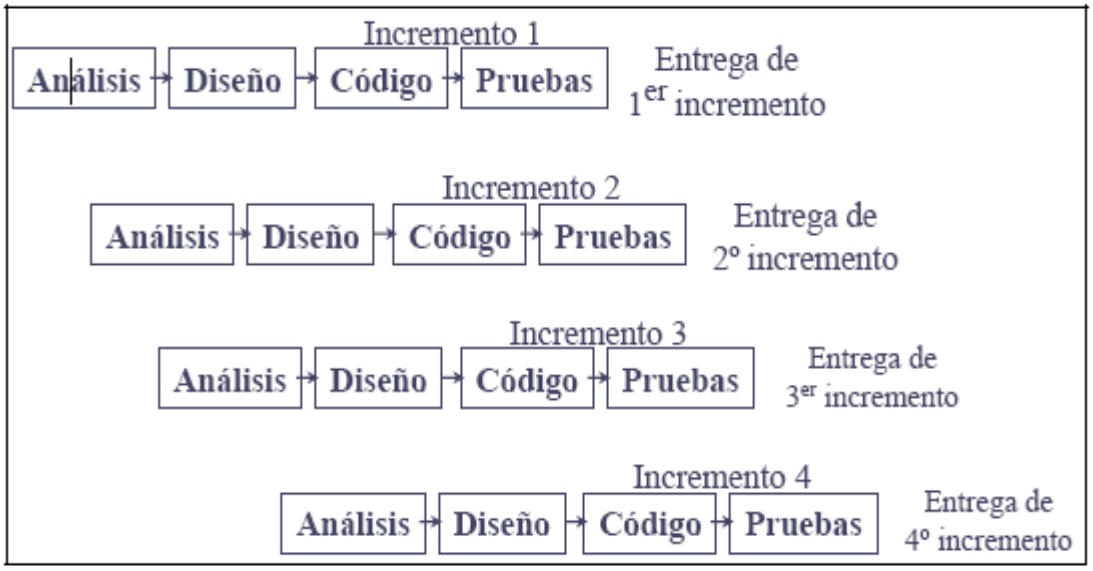

Figura 2.4 Modelo incremental

Cuando se utiliza un modelo incremental, el primer incremento a menudo es un producto esencial (núcleo). Es decir, se afrontan requisitos básicos, pero muchas funciones suplementarias (algunas conocidas) quedan sin extraer.

Como resultado de utilización y/o de evaluación, se desarrolla un plan para el incremento afrontando la modificación del producto central a fin de cumplir:
Las necesidades del cliente, la entrega de funciones, y características adicionales.

Este proceso se repite siguiendo la entrega de cada incremento, hasta que se elabore el producto completo. A diferencia de la construcción de prototipos, el modelo incremental se centra en la entrega de un producto operacional con cada incremento. Los primeros incrementos son versiones desmontadas del producto final, pero proporcionan la capacidad que sirve al usuario y también proporciona una plataforma para la evaluación por parte del usuario.

2.1.4 Método en espiral

El Proceso es representado como una espiral en lugar de una secuencia de actividades con retrocesos, cada giro en la espiral representa una fase en el proceso, no hay fases fijas tales como especificación o diseño (se gira en la espiral

dependiendo de qué se requiere). Los Riesgos son explícitamente identificados y resueltos durante el proceso. Para cada actividad habrá cuatro tareas (Figura 2.5):

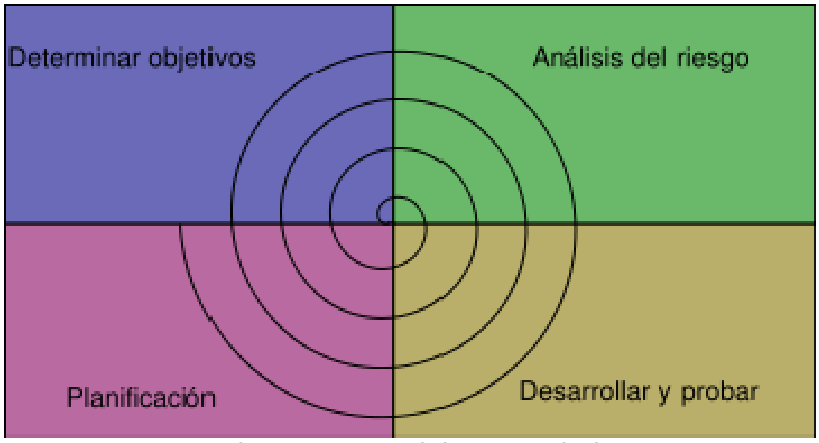

Figura 2.5 Modelo en espiral

Determinar o fijar objetivos

- Se identifican objetivos específicos para la fase.
- Fijar los productos definidos a obtener: requerimientos, especificaciones, manual de usuario.
- Fijar las restricciones.
- Identificación de riesgos del proyecto y estrategias alternativas para evitarlos.
- Hay una cosa que solo se hace una vez: planificación inicial o previa.

Análisis del riesgo

- Riesgos son identificados y se realizan actividades para reducir los riesgos clave.
- e estudian todos los riesgos potenciales y se seleccionan una o varias alternativas propuestas para reducir o eliminar los riesgos.

Desarrollo, verificar y validar

- Se escoge un modelo de desarrollo para el sistema que puede ser cualquiera de los modelos genéricos.
- Tareas de la actividad propia se prueban.

Planificar

- Se revisa el proyecto y se planifica la siguiente fase de la espiral.
- Revisamos todo lo hecho, evaluándolo, y con ello decidimos si continuamos con las fases siguientes y planificamos la próxima actividad.

Con cada iteración alrededor de la espiral (comenzando en el centro y siguiendo hacia el exterior), se construyen sucesivas versiones del software, cada vez más completas. Durante la primera vuelta alrededor de la espiral se definen los objetivos, las alternativas y las restricciones, y se analizan e identifican los riesgos. Si el análisis de riesgo indica que hay una incertidumbre en los requisitos, se puede usar la creación de prototipos en el cuadrante de ingeniería para dar asistencia tanto al encargado del desarrollo como al cliente. Se pueden usar simulaciones y otros modelos para definir más el problema y refinar los requisitos.

El cliente evalúa el trabajo de ingeniería (cuadrante de desarrollo y pruebas) y sugiere modificaciones. En base a los comentarios del cliente se produce la siguiente fase de planificación y de análisis de riesgo. En cada bucle alrededor de la espiral, la culminación del análisis de riesgo resulta en una decisión de “seguir o no seguir”.

Sin embargo, en la mayoría de los casos, se sigue avanzando alrededor del camino de la espiral, y ese camino lleva a los desarrolladores hacia fuera, hacia un modelo más completo del sistema, y al final, al propio sistema operacional. Cada vuelta alrededor de la espiral requiere ingeniería, que se puede llevar a cabo mediante el enfoque del ciclo de vida clásico o de la creación de prototipos. Debe tenerse en cuenta que el número de actividades de desarrollo que ocurren en el cuadrante inferior derecho aumenta al alejarse del centro de la espiral.

Como el software evoluciona, a medida que progresa el proceso, el desarrollador y el cliente comprenden y reaccionan mejor ante riesgos en cada uno de los niveles evolutivos

2.1.5 Modelo extreme programming (XP)

Es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación continua entre todos los participantes, simplicidad en las soluciones implementadas y facilidad para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos, muy cambiantes y donde existe un alto riesgo técnico.

Historia de usuario. Es la técnica utilizada en XP para especificar los requisitos del software. Se trata de tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer. Cada historia de usuario debe ser lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas. Posteriormente estas historias de usuario son descompuestas en tareas de programación y asignadas a los programadores para ser implementadas durante una iteración.

Roles XP

Programador. El programador escribe las pruebas unitarias y produce el código del sistema. Debe existir una comunicación y coordinación adecuada entre los programadores y otros miembros del equipo.

Cliente. El cliente escribe las historias de usuario y las pruebas funcionales para validar su implementación. Además, asigna la prioridad a las historias de usuario y decide cuáles se implementan en cada iteración centrándose en aportar mayor valor al negocio

Encargado de pruebas (*Tester*). El encargado de pruebas ayuda al cliente a escribir las pruebas funcionales. Ejecuta las pruebas, difunde los resultados en el equipo y es responsable de las herramientas de soporte para pruebas.

Encargado de seguimiento (*Tracker*). El encargado de seguimiento proporciona realimentación al equipo en el proceso XP. Realiza el seguimiento del progreso de

cada iteración y evalúa si los objetivos son alcanzables con las restricciones de tiempo y recursos presentes. Determina cuándo es necesario realizar algún cambio para lograr los objetivos de cada iteración.

Entrenador (*Coach*). Es responsable del proceso global. Es necesario que conozca a fondo el proceso XP para proveer guías a los miembros del equipo de forma que se apliquen las prácticas XP y se siga el proceso correctamente.

Consultor. Es un miembro externo del equipo con un conocimiento específico en algún tema necesario para el proyecto. Guía al equipo para resolver un problema específico.

Gestor (*Big boss*). Es el vínculo entre clientes y programadores, ayuda a que el equipo trabaje efectivamente creando las condiciones adecuadas. Su labor esencial es de coordinación.

Proceso XP

El proceso de desarrollo de XP, de manera general, consiste en los siguientes pasos:

a) El cliente define el valor de negocio a implementar.
b) El programador estima el esfuerzo necesario para su implementación.
c) El cliente selecciona qué construir, de acuerdo con sus prioridades y las restricciones de tiempo.
d) El programador construye ese valor de negocio.
e) Vuelve al primer paso.

El ciclo de vida ideal de XP está formado por seis fases: Exploración, Planificación de la Entrega (*Release*), Iteraciones, Producción, Mantenimiento y Muerte del Proyecto.

Fase I Exploración. Los clientes plantean a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto.

Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo. La fase de exploración toma de pocas semanas a pocos meses, dependiendo del tamaño y familiaridad que tengan los programadores con la tecnología.

Fase II Planificación de la entrega. El cliente establece la prioridad de cada historia de usuario, y correspondientemente, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. Una entrega debería obtenerse en no más de tres meses. Esta fase dura unos pocos días.

Fase III Iteraciones. Esta fase incluye varias iteraciones sobre el sistema antes de ser entregado. El plan de entrega está compuesto por iteraciones de no más de tres semanas. En la primera iteración se puede intentar establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto. Esto se logra escogiendo las historias que forzan la creación de esta arquitectura, sin embargo, esto no siempre es posible ya que es el cliente quien decide qué historias se implementarán en cada iteración. Los elementos que deben tomarse en cuenta durante la elaboración del Plan de la Iteración son:

a) Historias de usuario no abordadas.
b) Velocidad del proyecto.
c) Pruebas de aceptación no superadas en la iteración anterior.
d) Tareas no terminadas en la iteración anterior.

Todo el trabajo de la iteración es expresado en tareas de programación, cada una de ellas es asignada a un programador como responsable, pero llevadas a cabo por parejas de programadores.

Fase IV Producción. La fase de producción requiere de pruebas adicionales y revisiones de rendimiento antes de que el sistema sea trasladado al entorno del cliente. Al mismo tiempo, se deben tomar decisiones sobre la inclusión de nuevas

características a la versión actual, debido a cambios durante esta fase. Es posible que se rebaje el tiempo que toma cada iteración.

Fase V Mantenimiento. Mientras la primera versión se encuentra en producción, el proyecto XP debe mantener el sistema en funcionamiento al mismo tiempo que desarrolla nuevas iteraciones. Para realizar esto se requiere de tareas de soporte para el cliente. De esta forma, la velocidad de desarrollo puede bajar después de la puesta del sistema en producción.

Fase VI Muerte del proyecto. Es cuando el cliente no tiene más historias para ser incluidas en el sistema. Esto requiere que se satisfagan las necesidades del cliente en otros aspectos como rendimiento y confiabilidad del sistema. Se genera la documentación final del sistema y no se realizan más cambios en la arquitectura. La muerte del proyecto también ocurre cuando el sistema no genera los beneficios esperados por el cliente o cuando no hay presupuesto para mantenerlo.