UNIDAD 2

- **¿Cómo enfrentar el cambio?**
  - **Prototipos.**
  - **Entrega incremental.**
- **Organización de equipos y metodología ágil:**
  - **Manifiesto ágil.**
  - **XP (Programación Extrema). Historias de Usuario. Pruebas.**
  - **SCRUM. Roles y responsabilidades. Ciclo de trabajo.**

Organización de equipos y metodología ágil

Programación extrema

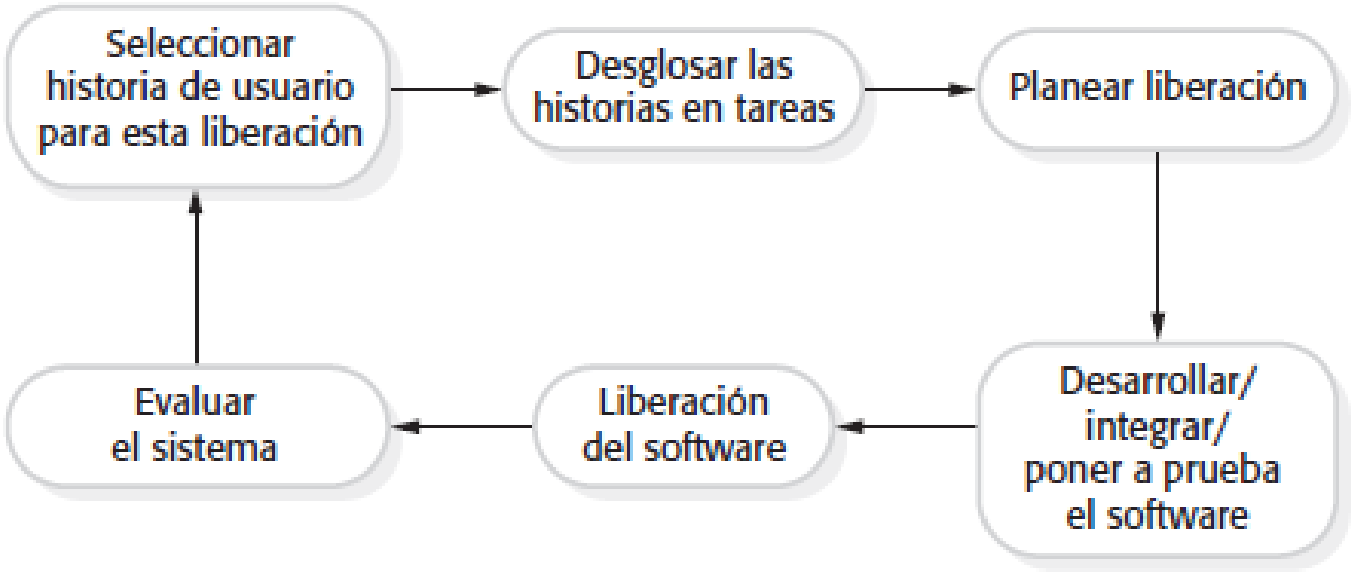
En la programación extrema, **los requerimientos se expresan como escenarios** (llamados **historias de usuario**), que se implementan directamente como una **serie de tareas**. Los programadores **trabajan en pares** y **antes de escribir el código desarrollan pruebas para cada tarea**.

Todas las pruebas deben ejecutarse con éxito una vez que el nuevo código se integre en el sistema.

Entre las liberaciones del sistema existe un breve lapso.

Organización de equipos y metodología ágil

Programación extrema



Organización de equipos y metodología ágil

Programación extrema

La programación extrema incluye algunas prácticas, las cuales reflejan los principios de los métodos ágiles:

1. El desarrollo incremental se apoya en pequeñas y frecuentes liberaciones del sistema. Los requerimientos se fundamentan en simples historias del cliente, o bien, en escenarios usados como base para decidir qué funcionalidad debe incluirse en un incremento del sistema.

2. La inclusión del cliente se apoya a través de un enlace continuo con el cliente en el equipo de desarrollo. El representante del cliente participa en el desarrollo y es responsable de definir las pruebas de aceptación para el sistema.

Organización de equipos y metodología ágil

Programación extrema

3. Las personas, no los procesos, se basan en la programación en pares, en la propiedad colectiva del código del sistema y en un proceso de desarrollo sustentable que no incluya jornadas de trabajo excesivamente largas.

4. El cambio se acepta mediante liberaciones regulares del sistema a los clientes, desarrollo de primera prueba, refactorización para evitar degeneración del código e integración continua de nueva funcionalidad.

5. Mantener la simplicidad se logra mediante la refactorización constante, que mejora la calidad del código, y con el uso de diseños simples que no anticipan innecesariamente futuros cambios al sistema.

Organización de equipos y metodología ágil

Programación extrema

Las tarjetas de historia son las entradas principales al proceso de planeación XP. Una vez diseñadas las tarjetas de historia, el equipo de desarrollo las descompone en tareas y estima el esfuerzo y los recursos requeridos para implementar cada tarea.

Esto involucra por lo general discusiones con el cliente para refinar los requerimientos. Entonces, para su implementación, el cliente prioriza las historias y elige aquellas que pueden usarse inmediatamente para entregar apoyo empresarial útil.

La intención es identificar funcionalidades útiles que puedan implementarse en aproximadamente dos semanas.

Organización de equipos y metodología ágil

Programación extrema

Conforme cambian los requerimientos, las historias no implementadas cambian o se desechan. Si se demandan cambios para un sistema que ya se entregó, se desarrollan nuevas tarjetas de historia y, otra vez, el cliente decide si dichos cambios tienen prioridad sobre la nueva función.

A veces, durante la planeación, salen a la luz preguntas que no pueden responderse fácilmente y se requiere trabajo adicional para explorar posibles soluciones. El equipo puede elaborar algún **prototipo** o tratar de desarrollarlo para entender el problema y la solución.

En términos XP, éste es un "pico" (spike), es decir, un incremento donde no se realiza programación. También suele haber "picos" para diseñar la arquitectura del sistema o desarrollar la documentación del sistema.

Nuevas versiones del software se construyen varias veces al día y las versiones se entregan a los clientes aproximadamente cada dos semanas.

Organización de equipos y metodología ágil

Programación extrema

Nunca se descuidan las fechas límite de las liberaciones; si hay problemas de desarrollo, se consulta al cliente y la funcionalidad se elimina de la liberación planeada.

Cuando un programador diseña el sistema para crear una nueva versión, debe correr todas las pruebas automatizadas existentes, así como las pruebas para la nueva funcionalidad.

La nueva construcción del software se acepta siempre que todas las pruebas se ejecuten con éxito. Entonces esto se convierte en la base para la siguiente iteración del sistema.

Organización de equipos y metodología ágil

Programación extrema

Un precepto fundamental de la ingeniería de software tradicional es que se tiene que diseñar para cambiar. Esto es, deben anticiparse cambios futuros al software y diseñarlo de manera que dichos cambios se implementen con facilidad. Sin embargo, la programación extrema descartó este principio basado en el hecho de que al diseñar para el cambio con frecuencia se desperdicia esfuerzo. No vale la pena gastar tiempo en adicionar generalidad a un programa para enfrentar el cambio. Los cambios anticipados casi nunca se materializan y en realidad pueden hacerse peticiones de cambio diametralmente opuestas.

Por lo tanto, el enfoque XP acepta que los cambios sucederán y cuando éstos ocurran realmente se reorganizará el software.

Organización de equipos y metodología ágil

Programación extrema

Un problema general con el desarrollo incremental es que tiende a degradar la estructura del software, de modo que los cambios al software se vuelven cada vez más difíciles de implementar. En esencia, el desarrollo avanza al encontrar soluciones alternativas a los problemas, con el resultado de que el código se duplica con frecuencia, partes del software se reutilizan de forma inadecuada y la estructura global se degrada conforme el código se agrega al sistema.

La programación extrema aborda este problema al sugerir que **el software debe refactorizarse continuamente**.

Esto significa que el equipo de programación busca posibles mejoras al software y las implementa de inmediato. Cuando un miembro del equipo observa un código que puede optimarse, realiza dichas mejoras, aun en situaciones donde no hay necesidad apremiante de ellas. Los ejemplos de refactorización incluyen la reorganización de una jerarquía de clases para remover un código duplicado, el ordenamiento y el cambio de nombre de atributos y métodos

Organización de equipos y metodología ágil

Programación extrema

Los entornos de desarrollo del programa, como Eclipse, incluyen herramientas para refactorizar, lo cual simplifica el proceso de encontrar dependencias entre secciones de código y realizar modificaciones globales al código.

Entonces, en principio, el software siempre debe ser de fácil comprensión y cambiar a medida que se implementen nuevas historias. En la práctica, no siempre es el caso. En ocasiones la presión del desarrollo significa que la refactorización se demora, porque se dedica el tiempo a la implementación de una nueva funcionalidad. Algunas características y cambios nuevos no pueden ajustarse con facilidad al refactorizar el nivel del código y al requerir modificar la arquitectura del sistema.

En la práctica, muchas compañías que adoptaron XP no usan todas las prácticas de programación extrema. Seleccionan según sus formas específicas de trabajar. Por ejemplo, algunas compañías encuentran útil la programación en pares; otras prefieren usar la programación y las revisiones individuales. Algunos programadores no hacen refactorización en partes del sistema que ellos no desarrollan, y pueden usarse requerimientos convencionales en vez de historias de usuario. Sin embargo, la mayoría de las compañías que adoptan una variante XP usan liberaciones pequeñas, desarrollo de primera prueba e integración continua.

Una Historia de la Prescripción de medicamentos

Una médica quiere prescribir fármacos a un paciente que se atiende en una clínica. El archivo del paciente ya se desplegó en su computadora, de manera que da clic en el campo del medicamento y luego puede seleccionar "medicamento actual", "medicamento nuevo" o "formulario".

Si selecciona "medicamento actual", el sistema le pide comprobar la dosis. Si quiere cambiar la dosis, ingresa la dosis y luego confirma la prescripción.

Si elige "medicamento nuevo", el sistema supone que la médica sabe cuál medicamento prescribir. Ella teclea las primeras letras del nombre del medicamento. El sistema muestra una lista de medicamentos posibles cuyo nombre inicia con dichas letras. Posteriormente elige el fármaco requerido y el sistema responde solicitándole que verifique que el medicamento seleccionado sea el correcto. Ella ingresa la dosis y luego confirma la prescripción.

Si elige "formulario", el sistema muestra un recuadro de búsqueda para el formulario aprobado. Entonces busca el medicamento requerido. Ella selecciona un medicamento y el sistema le pide comprobar que éste sea el correcto. Luego ingresa la dosis y confirma la prescripción.

El sistema siempre verifica que la dosis esté dentro del rango aprobado. Si no es así, le pide que la modifique.

Después de que ella confirme la prescripción, se desplegará para su verificación. La médica hace clic en "OK" o en "Cambiar". Si hace clic en "OK", la prescripción se registra en la base de datos de auditoría. Si hace clic en "Cambiar", reingresa al proceso de "prescripción de medicamento".

Prescripción de medicamentos

Tarea 1: Cambiar dosis del medicamento prescrito

Tarea 2: Selección de formulario

Tarea 3: Verificación de dosis

La verificación de dosis es una prevención de seguridad para comprobar que el médico no prescribe una dosis riesgosamente pequeña o grande.

Al usar el ID del formulario para el nombre genérico del medicamento, busca el formulario y recupera las dosis, máxima y mínima, recomendadas.

Verifica la dosis prescrita contra el mínimo y el máximo. Si está fuera de rango, emite un mensaje de error señalando que la dosis es muy alta o muy baja.

Si está dentro del rango, habilita el botón "Confirmar".

Organización de equipos y metodología ágil

Programación extrema

Pruebas en XP

Con el desarrollo incremental, no hay especificación de sistema que pueda usar un equipo de prueba externo para desarrollar pruebas del sistema.

En consecuencia, algunos enfoques del desarrollo incremental tienen un proceso de pruebas muy informal, comparado con las pruebas dirigidas por un plan.

Para evitar varios de los problemas de prueba y validación del sistema, XP enfatiza la importancia de la prueba de programa. La XP incluye un enfoque para probar que reduce las posibilidades de introducir errores no detectados en la versión actual del sistema.

Organización de equipos y metodología ágil

Programación extrema

Las características clave de poner a prueba en XP son:

1. desarrollo de primera prueba,
2. desarrollo de pruebas incrementales a partir de escenarios,
3. involucramiento del usuario en el desarrollo y la validación de pruebas, y
4. el uso de marcos de pruebas automatizadas.

El desarrollo de la **primera prueba** es una de las innovaciones más importantes en XP. En lugar de escribir algún código y luego las pruebas para dicho código, **las pruebas se elaboran antes de escribir el código.**

Esto significa que la prueba puede correrse conforme se escribe el código y descubrir problemas durante el desarrollo.

Organización de equipos y metodología ágil

Programación extrema

El papel del cliente en el proceso de pruebas es ayudar a desarrollar pruebas de aceptación para las historias, que deban implementarse en la siguiente liberación del sistema.

Las pruebas de aceptación son el proceso donde el sistema se pone a prueba usando datos del cliente para verificar que se cubren las necesidades reales de éste.

Para la historia del ejemplo, la prueba de aceptación implicaría escenarios donde

a) se cambió la dosis de un medicamento,
b) se seleccionó un nuevo medicamento y
c) se usó el formulario para encontrar un medicamento.

En la práctica, se requiere por lo general una serie de pruebas de aceptación en vez de una sola prueba.

Organización de equipos y metodología ágil

Programación extrema

Prueba 4: Comprobación de dosis

Entrada:
1. Un número en mg que represente una sola dosis del medicamento.
2. Un número que signifique el número de dosis individuales por día.

Pruebas:
1. Probar las entradas donde la dosis individual sea correcta, pero la frecuencia muy elevada.
2. Probar las entradas donde la dosis individual sea muy alta y muy baja.
3. Probar las entradas donde la dosis individual × frecuencia sea muy alta y muy baja.
4. Probar las entradas donde la dosis individual × frecuencia esté en el rango permitido.

Salida:
OK o mensaje de error que indique que la dosis está fuera del rango de seguridad.

Organización de equipos y metodología ágil

Programación extrema

Programación en pares

Otra práctica innovadora que se introdujo en XP es que los programadores trabajan en pares para desarrollar el software.

En realidad, trabajan juntos en la misma estación de trabajo para desarrollar el software.

Sin embargo, los mismos pares no siempre programan juntos. En vez de ello, los pares se crean dinámicamente, de manera que todos los miembros del equipo trabajen entre sí durante el proceso de desarrollo.

Desarrollo ágil de software

Programación en pares, ventajas:

1. Apoya la idea de la propiedad y responsabilidad colectivas para el sistema.
2. Actúa como un proceso de revisión informal, porque al menos dos personas observan cada línea de código.
3. Ayuda a la refactorización, que es un proceso de mejoramiento del software. Donde se usan la programación en pares y la propiedad colectiva, otros se benefician inmediatamente de la refactorización, de modo que es probable que apoyen el proceso.

Escalamiento de métodos ágiles

Los métodos ágiles se desarrollaron para usarse en pequeños equipos de programación, que podían trabajar juntos en la misma habitación y comunicarse de manera informal. Por lo tanto, los métodos ágiles se emplean principalmente para el diseño de sistemas pequeños y medianos. Desde luego, la necesidad de entrega más rápida del software, que es más adecuada para las necesidades del cliente, se aplica también a sistemas más grandes.

Por consiguiente, hay un enorme interés en escalar los métodos ágiles para enfrentar los sistemas de mayor dimensión, desarrollados por grandes organizaciones.

Mas información en página 74 Ingeniería de Software Somerville 9na.