Metodologías Agiles

Metodologías agiles

A Principios de la década del ’90 surge un enfoque, demasiado revolucionario para la época, ya que iba en contra de toda creencia de que mediante procesos altamente definiéndose iba a lograr obtener en tiempo, costo y con la calidad requerida.

Martin, propone RAD(Rapid Application Development) que consistía en un entorno de desarrollo altamente productivo, en el que participaban grupos pequeños de programadores utilizando herramientas que generaban código en forma automática.

En 1996, Beck comienza con un desarrollo basado únicamente en sus experiencias, este desarrollo es puesto en operación cumpliendo el calendario estipulado. Como consecuencia de dicho proyecto, Kent Beck dio origen a XP iniciando el movimiento de metodologías agiles al que se anexarían otras metodologías que surgieron con anterioridad.

Durante el año 2001 se conforma un comité de 17 expertos en la industria del software, con el objetivo de esbozar los principios que deberían permitir a los equipos software rápidamente y respondiendo a los cambios que puedan surgir a lo largo del proyecto, de aquí nace formalmente el término ágil aplicado al desarrollo de software. La idea era poder 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.

Tras esta reunión se creó The Agile Alliance, una organización, sin fines de lucro, dedicada a promover los conceptos relacionados con el desarrollo ágil de software y ayudar a los organizadores que adopten dichos puntos. El punto de partida fue el Manifiesto Ágil, un documento que resume la filosofía “ágil”.

Manifiesto por el Desarrollo Ágil de Software

Estamos descubriendo formas mejores de desarrollar software tanto por nuestra propia experiencia como ayudando a terceros. A través de este trabajo hemos aprendido a valorar:

- 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

Esto es, aunque valoramos los elementos de la derecha, valoramos más los de la izquierda

Principios del Manifiesto Ágil

Seguimos estos principios:

Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor.

Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos Ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.

Entregamos software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible.

Los responsables de negocio y los desarrolladores trabajamos juntos de forma cotidiana durante todo el proyecto.

Los proyectos se desarrollan en torno a individuos motivados. Hay que darles el entorno y el apoyo que necesitan, y confiarles la ejecución del trabajo.

El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.

El software funcionando es la medida principal de progreso.

Los procesos Ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante de forma indefinida.

La atención continua a la excelencia técnica y al buen diseño mejora la Agilidad.

La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial.

Las mejores arquitecturas, requisitos y diseños emergen de equipos auto-organizados.

A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo para a continuación ajustar y perfeccionar su comportamiento en consecuencia.

Metodologías agiles VS metodologías tradicionales

Metodología Agiles	Metodologías Tradicionales
Basadas en heurísticas provenientes de prácticas de producción de código	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo
Especialmente preparas par cambios durante el proyecto	Cierta resistencia a los cambios
Impuestas internamente (por el equipo)	Impuestas externamente
Procesos menos controlados, con pocos principios	Proceso mucho más controlado, con numerosas políticas/normas.
No existe el contrato tradicional o al menos es bastante flexible	Existe un contrato prefijado
El cliente es parte del equipo de desarrollo	El cliente interactúa con el equipo de desarrollo mediante reuniones
Grupos pequeños (<10 integrantes) y trabajando en el mismo sitio.	Grupos grandes y posiblemente distribuidos
Pocos artefactos	Mas artefactos
Pocos roles	Mas roles
Menos énfasis en la arquitectura del software	La arquitectura del software es esencial y se expresa mediante modelos

XP- eXtreme Programming

XP es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en la realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe el alto riesgo técnico.

Características esenciales

A. Las historias de usuario

Son la técnica utilizada para especificar los requisitos del software. Se trata de tarjetas en papel en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. El tratamiento de las historias de usuario es muy dinámico y flexible. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas.

Ej Contenido:

- Fecha
- Tipo de Actividad (nueva, Corrección, mejora)
- Prueba Funcional
- Numero de historia
- Prioridad técnica
- Prioridad del cliente
- Referencia a otra historia previa
- Riesgo
- Estimación técnica

- Descripción
- Notas
- Lista de seguimiento con fecha
- Estado de pendientes
- Comentarios

Al momento de planificar, es necesarios, tener en cuenta que las historias deben durar entre una y tres semanas en tiempo de programación. Cada una de las historias es descompuesta en tareas de programación y asignadas a los programadores para ser implementadas durante una iteración

B. Roles

- **Programador**: escribe las pruebas unitarias y produce el código del sistema
- **Cliente**: Escribe las historias de usuario y las pruebas funcionales para valida su implementación. además, asigna las prioridades a las historias de usuario y decide cuales se implementan en cada iteración centrándose en aportar mayor valor al negocio
- **Encargado de pruebas**: Ayuda al cliente a escribir las pruebas funcionales. Ejecuta las pruebas regularmente, difunde los resultados en el equipo y es responsable de las herramientas de soporte para pruebas
- **Encargado de seguimiento**: Proporciona realimentación al equipo. Verifica el frado de acierto entre las estimaciones realizadas y el tiempo real dedicado, para mejorar en futuras estimaciones. Realiza el seguimiento del progreso de cada iteración.
- **Entrenador**. Es responsable del proceso global. Debe proveer guías al equipo de forma que se apliquen las practicas 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, en el que puedan surgir problemas
- **Gestor**. Es el vínculo entre cliente y programadores, ayuda a que el equipo trabaje efectivamente creando las condiciones adecuadas. Su labor esencial es de coordinación

C. Proceso

1. El cliente define el valor de negocio a implementar.
2. El programador estima el esfuerzo necesario para su implementación
3. El cliente selecciona que construir, de acuerdo con sus prioridades y las restricciones de tiempo
4. El programador construye ese valor de negocio
5. Vuelve al paso 1

D. Practicas XP

La principal suposición que se realiza en XP es la posibilidad de disminuir la mítica curva exponencial del costo del cambio a lo largo del proyecto, lo suficiente para que el diseño evolutivo funcione. Esto se consigue gracias a las tecnologías disponibles para ayudar en el desarrollo de software y a la aplicación disciplinada de las siguientes practicas

- Planificación
- Entregas pequeñas
- Historias en común con el cliente
- Diseño Simple
- Pruebas

- Refactorización (constate ordenamiento del código)
- Programación en parejas
- Propiedad colectiva del código
- Integración continua
- 40HS por semana
- Cliente in situ
- Estándares de programación

Referencias

Metodologías Agiles – Amaro Calderon, Sarah Damaris, Valverde Rebaza

http://agilemanifesto.org/iso/es/