Metodologías de Desarrollo Java EE

Sesión 1: Metodologías Ágiles

Puntos a tratar

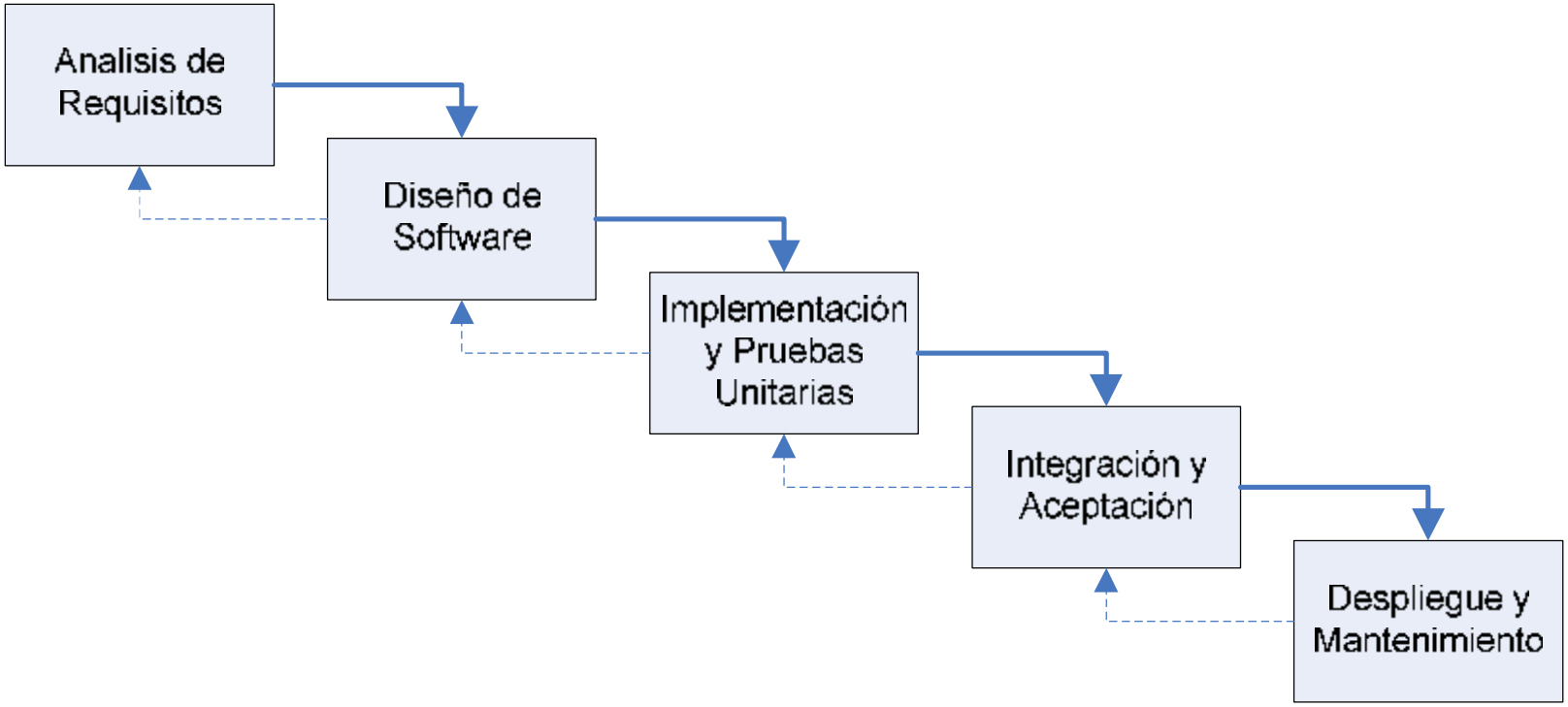
- Ciclo de Vida Clásico
- Desarrollo Iterativo e Incremental
- Manifiesto Ágil
- Principios Ágiles
- Conceptos Ágiles
- Entorno Ágil
- Agilidad vs Disciplina
- Metodologías Ágiles
- Roadmap

Introducción

- Las nuevas metodologías deben dejar de lado los formalismos de metodologías clásicas para guiarse por los resultados a corto/medio plazo, pero sin perder de vista la calidad.
- Existe una tendencia a la perdida de la formalidad y el rigor, buscando el equilibrio entre una documentación extensa respecto a una documentación útil.
- Durante los últimos 5 años han ido apareciendo diferentes metodologías, conocidas como *ágiles*, que dan mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental del software con iteraciones muy cortas.

Ciclo de Vida Clásico

- *Modelo secuencial o en cascada*



Fases del Ciclo de Vida Clásico

1. **Análisis de los requisitos**: Se establecen los servicios, restricciones y objetivos con los usuarios del sistema. Se busca hacer esta definición en detalle.
2. **Diseño de software**: Se divide el sistema en sistemas de software o hardware. Se establece la arquitectura total del sistema. Se identifican y describen las abstracciones y relaciones de los componentes del sistema.
3. **Implementación y pruebas unitarias**: Construcción de los módulos y unidades de software. Se realizan pruebas de cada unidad.
4. **Integración y pruebas del sistema**: Se integran todas las unidades. Se prueban en conjunto. Se entrega el conjunto probado al cliente.
5. **Despliegue y mantenimiento**: Generalmente es la fase más larga. El sistema es puesto en marcha y se realiza la corrección de errores descubiertos. Se realizan mejoras de implementación. Se identifican nuevos requisitos.

Problemas del Ciclo de Vida Clásico

- Las iteraciones son costosas e implican rehacer trabajo debido a la producción y aprobación de documentos.
- Aunque son pocas iteraciones, es normal congelar parte del desarrollo y continuar con las siguientes fases.
- Los problemas se dejan para su posterior resolución, lo que lleva a que estos sean ignorados o corregidos de una forma poco elegante.
- Existe una alta probabilidad de que el software no cumpla con los requisitos del usuario por el largo tiempo de entrega del producto.
- Es inflexible a la hora de evolucionar para incorporar nuevos requisitos. Es difícil responder a cambios en los requisitos.

Realidad

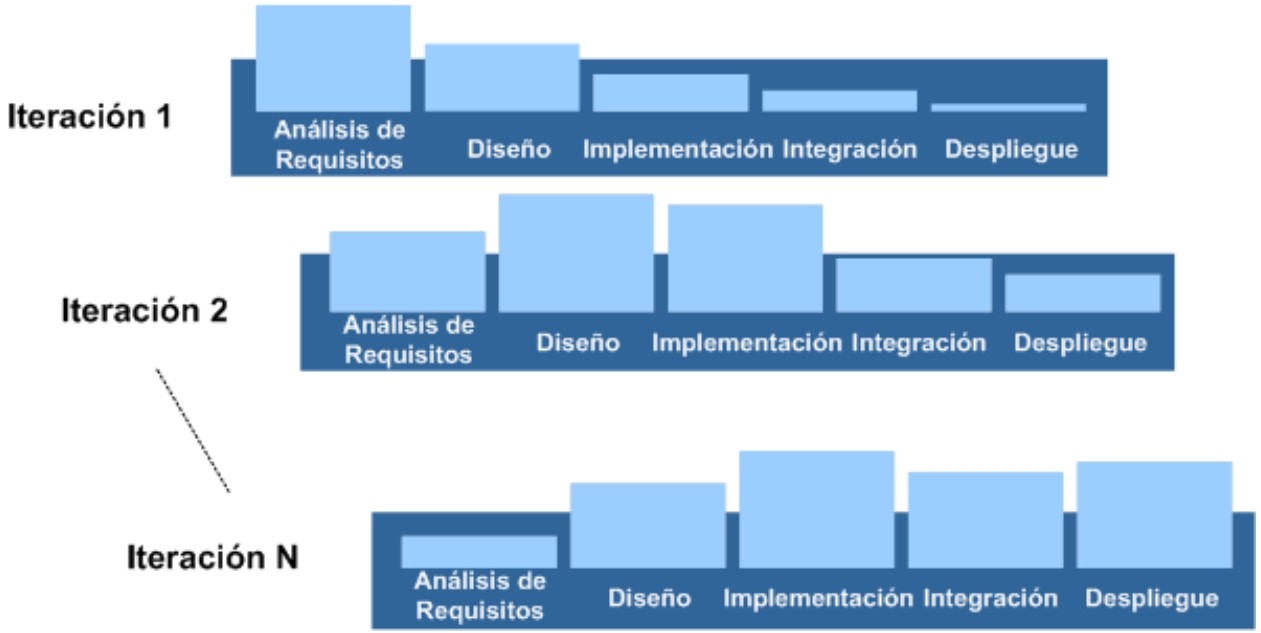
La única **constante** en los proyectos de software es el **cambio**

Desarrollo Iterativo I

- Implica la programación y prueba de sistemas parciales mediante *ciclos repetitivos*, de modo que los cambios son "fácilmente" asimilados
- El desarrollo comienza incluso antes de que todos los requisitos hayan sido detallados.
- El elemento clave para aclarar y refinar las especificaciones, que no dejan de cambiar y/o evolucionar, es la *retroalimentación*.
- Las rápidas iteraciones de desarrollo, la retroalimentación y adaptación del software permiten refinar los requisitos y el diseño.

Desarrollo Iterativo II

- El resultado de cada iteración es un trozo de sistema que está probado, integrado y *que se puede utilizar*
- Cada iteración incluye su propias fases de análisis de requisitos, diseño, implementación y prueba.



Iteraciones *Timeboxed*

- Iteraciones donde se fija la duración y no se permite modificarla.
- La duración total de un proyecto también puede ser *timeboxed*.
- Si finalmente llegamos a la conclusión de que las peticiones (el ámbito) para la iteración no se pueden implementar dentro del timebox
  - en vez de alargar la finalización de la iteración
  - se debe reducir el ámbito (recolocando las peticiones con menor prioridad de nuevo en la lista de peticiones)
  - para que el sistema parcial siga creciendo y siempre finalice en un estado estable y probado en la fecha final de la iteración originalmente planeada.

Movimiento Ágil

- Los métodos de desarrollo ágil promueven
  - desarrollo evolutivo
  - plazos de entrega con plazos cortos y *timeboxed*
  - planificación dinámica y adaptativa,
  - entregas incrementales,
  - prácticas que incentivan la agilidad

    respuesta rápida y flexible al cambio.

- El lema de los métodos ágiles es

"abrazar los cambios“

(*embrace change*)

Agile Alliance

- En 2001 un grupo de expertos interesados en los método iterativos y ágiles se reunieron para encontrar un fondo común.
- De aquí nació la Alianza Ágil (www.agilealliance.com) con un manifiesto y una serie de principios.
- El manifiesto y los principios han sido, y son, la base de las metodologías ágiles
  - la gestión ágil de proyectos, el modelado ágil, las técnicas ágiles, etc... son una consecuencia de dicha reunión.

Manifiesto Ágil

- **Individualidades e interacciones**

  frente a procesos y herramientas
- **Software que funciona**

  frente documentación comprensible
- **Colaboración del cliente**

  frente a la negociación de un contrato
- **Respuesta al cambio**

  frente al seguimiento estricto de un planning

www.agilemanifesto.org

Individualidades e interacciones...

... frente a procesos y herramientas

- La gente, y su interacción, es el principal factor de éxito de un proyecto software.
  - Si se sigue un buen proceso de desarrollo, pero el equipo falla, el éxito no está asegurado
  - Si el equipo funciona, es más fácil conseguir el objetivo final, aunque no se tenga un proceso bien definido.
- Es más importante construir un buen equipo que construir el entorno.
  - Muchas veces se comete el error de construir primero el entorno y esperar que el equipo se adapte automáticamente.
  - Es mejor crear el equipo y que éste configure su propio entorno de desarrollo en base a sus necesidades.

Software que funciona...

... frente documentación comprensible

- Partiendo de que el software sin documentación es un desastre, la regla a seguir es:
  - **no producir documentos a menos que sean necesarios de forma inmediata para tomar un decisión importante**
- Demasiada documentación es peor que poca
- Los 2 mejores documentos para transferir la información a nuevos integrantes son el código fuente y el propio equipo
- Es el software producido por el equipo lo que utilizará el usuario, y no la documentación

Colaboración del cliente...

... frente a la negociación de un contrato

- Un contrato que especifica los requisitos, la planificación y el coste del proyecto falla por su base.
  - en la mayoría de los casos, los términos especificados pierden todo el sentido antes de que el proyecto se complete.
- Los mejores contratos son aquellos que determinan que el cliente y el equipo de desarrollo deben trabajan juntos.
- Los proyectos exitosos implican la retroalimentación del cliente a un ritmo constante.

Respuesta al cambio...

... frente al seguimiento de un planning estricto

- La habilidad de responder a los cambios determina el éxito del proyecto.
- El curso de un proyecto software no se puede predecir a largo plazo
  - existen demasiadas variable a tener en cuenta.
  - no somos suficientemente buenos para estimar el coste de un proyecto a largo plazo
- Solución = resolución descendiente de la planificación
  - planificaciones detalladas para las siguientes semanas (saber)
  - planificaciones menos detalladas para los siguientes meses (conocer)
  - planificaciones rudimentarias para las posteriores (ligera idea)

Principios Ágiles

- Son 12 características que diferencian un proceso ágil de uno tradicional.
- El objetivo de estos principios se divide en 2:
  - Ayudar a las personas a que comprendan mejor de que trata el desarrollo de software ágil
  - Determinar si un desarrollador esta siguiendo una metodología ágil o no
- Cabe destacar que estos principios no especifican un método
  - definen un conjunto de guías que debe cumplir cualquier enfoque que quiera estar bajo el paraguas "ágil"

Principios Ágiles I

1. La prioridad absoluta es satisfacer al cliente cuanto antes mejor, y de forma continuada entregar software útil que le reporte valor.
   - Un proceso es ágil si a las pocas semanas de empezar ya entrega software que funcione aunque sea rudimentario.
   - El cliente decide si pone en marcha dicho software con la funcionalidad que ahora le proporciona o simplemente lo revisa e informa de posibles cambios a realizar.

Principios Ágiles II

2. Apreciar los cambios de requisitos, incluso en fases tardías del desarrollo. Los procesos ágiles aprovechan los cambios para que el cliente tome ventaja competitiva.
   - Actitud que deben adoptar los miembros del equipo de desarrollo.
   - Los cambios en los requisitos deben verse como algo positivo.
     - Les va a permitir aprender más, a la vez que logran una mayor satisfacción del cliente
   - Implica que la estructura del software debe ser flexible para poder incorporar los cambios sin demasiado coste añadido. El paradigma orientado a objetos puede ayudar a conseguir esta flexibilidad.

Principios Ágiles III

3. Entregar frecuentemente software que funcione, desde un par de semanas a un par de meses, con el menor intervalo de tiempo posible entre una entrega y la siguiente
   - Las entregas al cliente se insiste en que sean software, no planificaciones, ni documentación de análisis o de diseño.
4. La gente del negocio (el cliente) y los desarrolladores deben trabajar juntos a lo largo del proyecto
   - El proceso de desarrollo necesita ser guiado por el cliente, por lo que la interacción con el equipo es muy frecuente.

Principios Ágiles IV

5. Construir proyectos en torno a individuos motivados. Darles el entorno y el apoyo que necesitan y confiar en ellos para conseguir el trabajo
   - La gente es el principal factor de éxito, todo los demás (proceso, entorno, gestión, etc.) queda en segundo plano.
   - Si cualquiera de ellos tiene un efecto negativo sobre los individuos debe ser cambiado.
6. El diálogo cara a cara es el método más eficiente y efectivo para comunicar información dentro de un equipo de desarrollo
   - Los miembros de equipo deben hablar entre ellos, éste es el principal modo de comunicación.
   - Se pueden crear documentos pero no todo estará en ellos, no es lo que el equipo espera.

Principios Ágiles V

7. La medida principal de progreso es el software que funciona
   - El estado de un proyecto no viene dado por la documentación generada o la fase en la que se encuentre, sino por el código generado y en funcionamiento.
     - Un proyecto se encuentra al 50% si el 50% de los requisitos ya están en funcionamiento.
8. Los procesos ágiles promueven un desarrollo sostenible. Los promotores, desarrolladores y usuarios deberían ser capaces de mantener una paz constante.
   - No se trata de desarrollar lo más rápido posible, sino de mantener el ritmo de desarrollo durante toda la duración del proyecto, asegurando en todo momento que la calidad de lo producido es máxima.

Principios Ágiles VI

9. La atención continua a la calidad técnica y al buen diseño aumentan la agilidad
   - Producir código claro y robusto es la clave para avanzar más rápidamente en el proyecto.
10. La simplicidad (arte de maximizar la cantidad de trabajo no realizado) es esencial
   - Tomar los caminos más simples que sean consistentes con los objetivos perseguidos.
   - Si el código producido es simple y de alta calidad será más sencillo adaptarlo a los cambios que puedan surgir.

Principios Ágiles VII

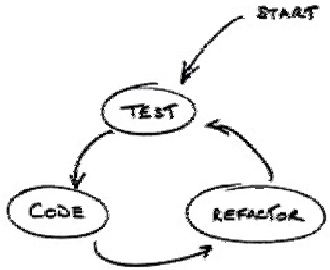
11. Las mejores arquitecturas, requisitos y diseños surgen de los equipos de trabajo organizados por sí mismos.
    - Todo el equipo es informado de las responsabilidades y éstas recaen sobre todos sus miembros.
    - Es el propio equipo el que decide la mejor forma de organizarse, de acuerdo a los objetivos que se persigan.
12. En intervalos regulares, el equipo reflexiona respecto a cómo llegar a ser más efectivo, y según esto ajusta su comportamiento.
    - Puesto que el entorno está cambiando continuamente, el equipo también debe ajustarse al nuevo escenario de forma continua.
    - Puede cambiar su organización, sus reglas, sus convenciones, sus relaciones, etc., para seguir siendo ágil.

Conceptos Ágiles I

- Abrazar los cambios
  - Afrontar los cambios como aliados (vs enemigos)
  - Los cambios permiten mayor creatividad y aportan valor más rápido
- Entregas frecuentes
  - Planificación de muchas entregas con periodos cortos.
  - Fuerza la implementación de las funciones de mayor prioridad, entregando valor al cliente rápidamente y emergen los requisitos.
- Diseño simple
  - Diseñar para la batalla, no para la guerra
  - Principio **KISS** (*Keep It Simple, Stupid*)
  - Principio **YAGNI** (*You Aren't Going to Need It*)
  - Realizar el diseño para cubrir el problema actualmente desarrollado.
  - Como los cambios son inevitables, planificar para funcionalidades futuras es malgastar esfuerzos.

Conceptos Ágiles II

- Refactorización
  - Reestructurar el software para eliminar la duplicación, mejorar la comunicación, simplificar, y añadir flexibilidad sin modificar el comportamiento.
  - Principio **DRY** (*Don't Repeat Yourself*)
  - Rediseño en caliente.



- Desarrollo Dirigido por las Pruebas
  - 1º escribir prueba, 2º escribir código, 3º ejecutar prueba
  - Las pruebas de los módulos y de los métodos se realizan incrementalmente por los desarrolladores y los clientes antes y durante la codificación.
  - Fomenta las iteraciones con ciclos cortos
- Programación en Parejas
  - 2 programadores trabajan codo con codo en un ordenador.
  - Colaboración continua en el mismo diseño, algoritmo, código y pruebas.

Conceptos Ágiles III

- Conocimiento Tácito
  - La agilidad se consigue estableciendo y actualizando el conocimiento del proyecto en las cabezas de los participantes.
  - Se evitan los documentos innecesarios (conocimiento explicito)
- Retrospectiva
  - Reunión post-iteración para evaluar la efectividad del trabajo realizado, métodos utilizados, y estimaciones
  - Fomenta el aprendizaje y mejora la estimación para futuras iteraciones

Entorno Ágil I

- Integraciones Continuas
  - Continua integración (CI) 24/7 de todo el código fuente del sistema.
  - En cuanto un cambio se sube al gestor de versiones, se construye el sistema, se despliega sobre el servidor y se ejecutan las pruebas.
  - En el caso de que las pruebas fallen, se pueden realizar diversas acciones
    - como enviar mails al arquitecto y al último desarrollador que ha subido el cambio al sistema.
  - Dentro de un proyecto Java, la media de construcción es de 15 a 30 minutos.
  - Nuestro sistema integra los cambios cuanto antes.

Entorno Ágil II

- Herramientas CASE e Ingeniería Inversa
  - Las herramientas CASE que soportan UML ofrecen:
    - Ingeniería Directa → generación de código a partir de los diagramas
    - Ingeniería Inversa → generación de diagramas a partir del código.
  - Dentro de los proyecto ágiles, las herramientas CASE se utilizan casi exclusivamente para la ingeniería inversa
    - ofreciendo mecanismos visuales para la comunicación entre los integrantes del equipo.
  - Las herramientas comerciales más utilizadas son Borland Together y Rational Software Architect; mixtas/gratuitas tenemos Omondo y Poseidon UML y StarUML.
- Wiki de Proyecto:
  - Permiten a los desarrolladores tener un repositorio compartido de conocimiento
  - Facilitan la comunicación de actividades y tareas.
  - Tenemos versiones sencillas como Usemod o más completas como XWiki o TWiki.

Entorno Ágil III

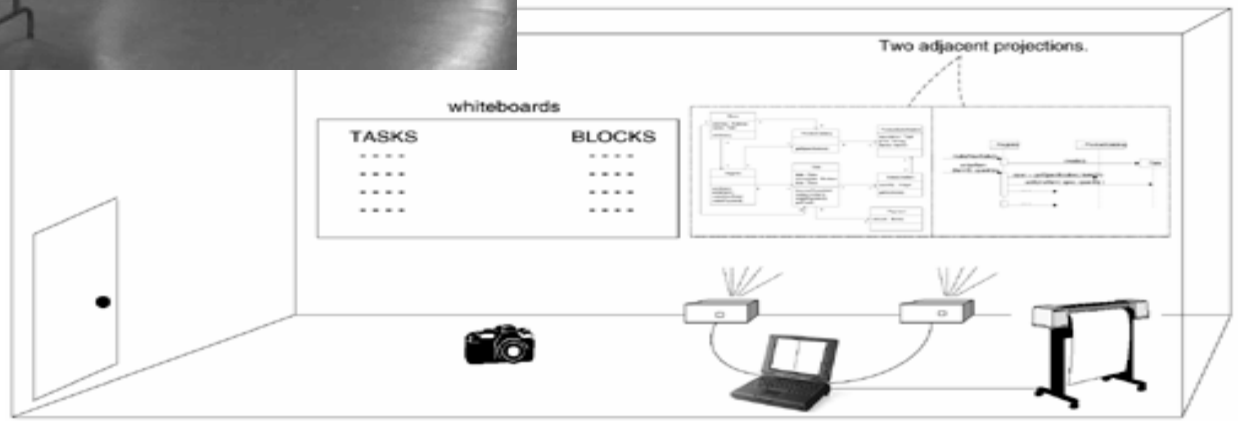
- Habitación Común
  - Todos los integrantes del equipo deben compartir una sala común
    - comunicación directa entre los desarrolladores y cliente.
  - Intimidad → pueden existir espacios privados separados
    - los desarrolladores pueden utilizar durante las tareas no relacionadas con el desarrollo.
  - Es aconsejable situar la mesas en el centro de la sala, para dejar las paredes libres

Entorno Ágil IV

- Paredes por Pizarras
  - El modelado ágil promueve utilizar pizarras velleda
    - necesitamos utilizar todo el espacio disponible (podemos situar las mesas en el centro de la sala).
  - Mediante las pizarras y rotuladores de pizarra, las paredes se convierten en otro mecanismo de comunicación.
- Cámaras Digitales y Proyectores
  - En vez de tener de tomar notas de los modelos diagramados en las paredes, una simple fotografía captura toda la información
    - nos permite imprimirla o publicarla en nuestro wiki.
  - Además, el uso de proyectores donde visualizar las fotografías previas y los diagramas obtenidos por ingeniería inversa también facilitan el trabajo.

Oficina Ágil

http://xp123.com/xplor/room-gallery



Agilidad vs. Disciplina

- Para comparar las metodologías ágiles y las orientadas a una planificación analizamos las características de :
  - Aplicación
  - Administración
  - Técnico
  - Personal

Características de la Aplicación

Característica	Ágil	Clásica
Objetivos Principales	Obtener valor rápida y continuamente, responder al cambio	Alta seguridad, predecible, repetible, optimizable
Tamaño	Equipo y proyecto pequeño/mediano	Equipo y proyecto grande
Entorno	Turbulentos, alta tasa de cambios, foco en el proyecto	Estables, pocos cambios, foco en proyecto y organización

Características de Gestión

Característica	Ágil	Clásica
Relación con el Cliente	Clientes in situ, dedicados al proyecto y centrado en priorizar requisitos	Participación del cliente sólo cuando se necesita y centrado en el contrato
Planificación y Control	La planificación es un medio para llegar al fin. Control cualitativo	Planificación para comunicar y coordinar. Control cuantitativo
Comunicación	Conocimiento tácito e interpersonal	Conocimiento explícito y documentado

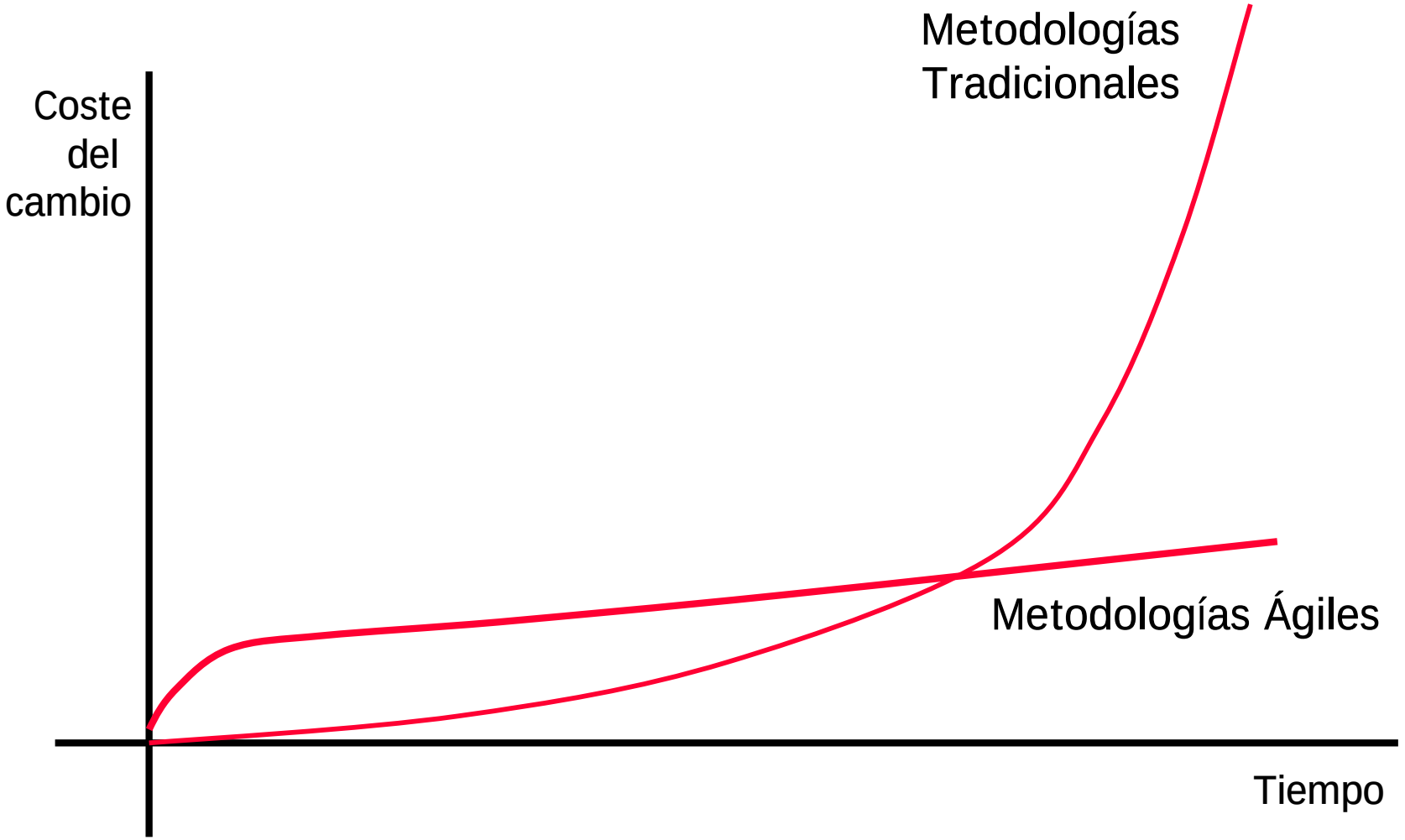
Características Técnicas

Característica	Ágil	Clásica
Requisitos	Historias informales y casos de prueba priorizados (valor + riesgo). Con cambios no predecibles	Especificaciones formales y completas bajo control de cambios. Requisitos no funcionales
Desarrollo	Diseño simple=YAGNI *(You Aren't Going to Need It)* Incrementos cortos. Refactorización barata.	Arquitectura para anticipar cambios. Incrementos mayores. Refactorización cara.
Pruebas	Pruebas antes de codificar. Incrementales. Casos de prueba ejecutables definen requerimientos	Plan y procedimientos de prueba a partir de las especificaciones.

Características Personales

Característica	Ágil	Clásica
Cliente	Dedicados y en el Lugar. CRACK *(collaborative, representative, authorized, committed, knowledgeable)*	CRACK sin fulltime
Desarrollador	Habilidades técnicas y sociales. Alto porcentaje de senior, el resto semi-senior	Perfiles menos talentosos. Alto porcentaje de senior al inicio, después perfiles distribuidos
Cultura	Reconocimiento a través de libertad y autonomía	Reconocimiento con políticas claras y procedimientos de roles

Comparando...


Metodologías
Tradicionales
Coste
del
cambio
Metodologías Ágiles
Tiempo

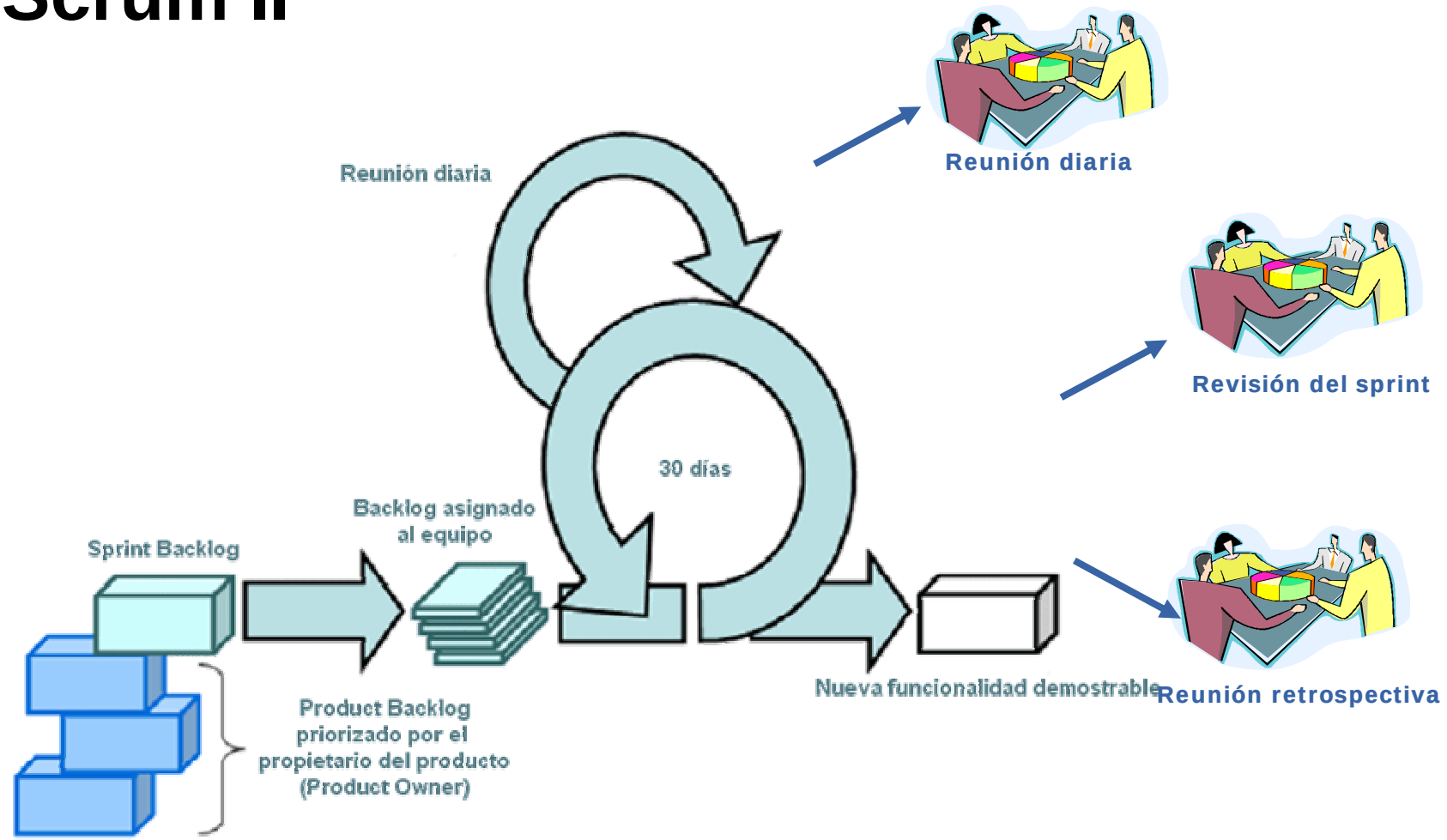
Metodología Ágil

- Una metodología es ágil cuando el desarrollo de software es:
  - **incremental** → entregas pequeñas de software, con ciclos rápidos
  - **cooperativo** → cliente y desarrolladores trabajan juntos constantemente con una cercana comunicación
  - **sencillo** → el método en sí mismo es fácil de aprender y modificar, bien documentado
  - **adaptable** → permite realizar cambios en cualquier momento

Scrum

- Desarrollada por *Ken Schwaber*, *Jeff Sutherland* y *Mike Beedle*.
- Define un marco para la gestión de proyectos promoviendo los equipos auto-organizados, con métricas de equipo diarias, y evitando la definición de pasos preestablecidos.
- Principales características:
  - Desarrollo de software mediante iteraciones de 30 dias, denominadas **Sprints**. El resultado de cada sprint es un incremento ejecutable que se muestra al cliente.
  - Reunión diaria de 15 minutos, denominada **Scrum**, del equipo de desarrollo, desarrollada de pie y en circulo, para coordinar e integrar el trabajo del dia a dia, así como comunicar los problemas encontrados.
  - Demo con el cliente al finalizar cada iteración.

Scrum II


Reunión diaria
Reunión diaria
Revisión del sprint
30 días
Backlog asignado
al equipo
Sprint Backlog
Nueva funcionalidad demostrable
Reunión retrospectiva
Product Backlog
priorizado por el
propietario del producto
(Product Owner)

Desarrollo de Software Adaptativo (ASD)

- Impulsado por Jim Highsmith
- Destaca el principio de que la adaptación continua del trabajo recién hecho es normal.
- El ciclo de vida que propone tiene tres fases repetitivas en ciclos:
  - *Especulación*: se inicia el proyecto y se planifican las características del software.
  - *Colaboración*: se desarrollan las características.
  - *Aprendizaje*: se revisa su calidad, y se entrega al cliente. La revisión de los componentes sirve para aprender de los errores y volver a iniciar el ciclo de desarrollo.
- Sus principales características son que está centrado en la misión, basado en características, iterativo, *timeboxed*, dirigido por los riesgos y tolerante a los cambios.

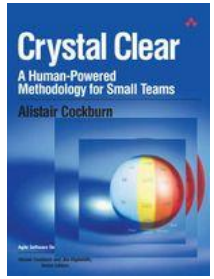
Lean Development (LD)

Lean
Software Development
An Agile Toolkit

- Definida por Bob Charette
- Parte de su experiencia en proyectos dentro del área de gestión de riesgos, y los principios y visión de los conceptos de **manufacturación *Lean*** (sin grasa).
- Los cambios se consideran riesgos, pero si se manejan adecuadamente se pueden convertir en oportunidades que mejoren la productividad del cliente.
  - Introduce un mecanismo para implementar dicha tolerancia a los cambios.
- No se limita únicamente al equipo de desarrollo, sino también involucra a todo el personal relacionado con el producto, como gestores.
- LD es más una estrategia de negocio y gestión de proyectos que un proceso de desarrollo
  - No especifica prácticas, políticas o guías de desarrollo.

Metodologías Crystal

- Conjunto de metodologías ágiles desarrolladas por *Alistair Cockburn*
- Pese a aceptar la necesidad de un ciclo de vida iterativo, destaca el desarrollo de software como **peopleware**
  - centrado en las personas que componen el equipo (de ellas depende el éxito del proyecto)
  - reducción al máximo del número de artefactos producidos.
- El desarrollo de software se considera un **juego cooperativo** de invención y comunicación
  - limitado por los recursos a utilizar.
- El equipo de desarrollo es un factor clave
  - invertir en mejorar sus habilidades y destrezas
  - definir políticas de trabajo en equipo.



- Las diferentes versiones de *Crystal* se nombran
  - con colores (*Clear*, *Yellow*, *Orange*, *Red*) para denotar el número de personas implicadas
  - la criticidad de los fallos en términos de pérdidas (Comfort, Discretionary Money, Essential Money, y Life).

Método de Desarrollo de Sistemas Dinámico (DSDM)

DSDM CONSORTIUM

- Nace en 1994 con el objetivo el objetivo de crear una metodología RAD unificada
- Proceso iterativo e incremental, donde el equipo de desarrollo y el usuario deben trabajar juntos.
- Ciclo de Vida con 5 fases
  - 1-Estudio Viabilidad, 2-Estudio del Negocio ← Secuenciales
  - 3-Modelado Funcional, 4-Diseño y Construcción, y 5-Implementación ← Iterativas e Incrementales
- Énfasis en las actividades de gestión de proyectos:
  - La planificación es necesaria en cada fase, conforme el plan evoluciona basado en los incrementos y sus resultados.
    - Se realizan planificaciones con contenidos detallados.
  - Las iteraciones son *timeboxed*, siendo el principal modo de planificar, monitorizar y controlar el proyecto.
    - El calendario y los costes se mantienen constantes, siendo variable la cantidad de requisitos a implementar.
  - Los requisitos se priorizan con la técnica **MoSCoW** (*Must have, Should Have, Could Have, Want*).

Proceso Unificado (UP)

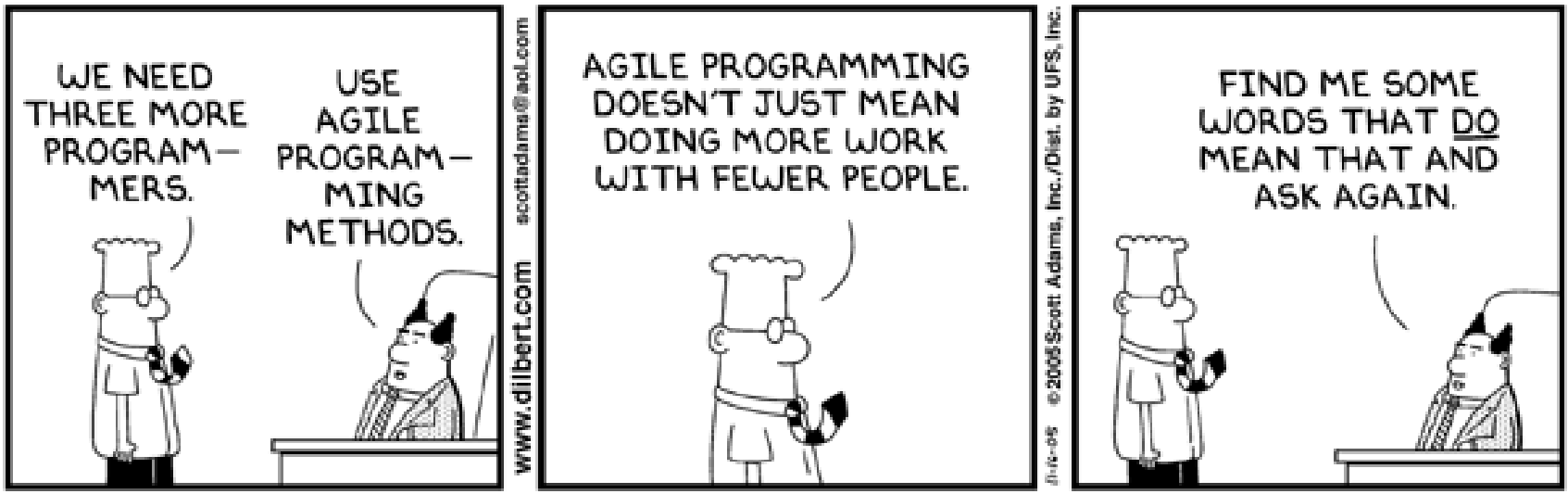
- Refinado a partir de la metodología RUP, ofrece una visión reducida, donde destaca:
  - Iteraciones cortas y *timeboxed*
  - Desarrollo de los elementos con mayor riesgo (*risk-driven*) y mayor valor en las primeras iteraciones, prefiriendo la reutilización de componentes existentes
  - Asegurar que se entrega valor al cliente
  - Afrontar los cambios desde el principio del proyecto
  - Trabajar juntos como un equipo
- Las iteraciones de UP, al igual que RUP, se agrupan en 4 fases:
  - inicio, elaboración (implementando los elementos de mayor riesgo y la arquitectura de la aplicación), construcción y transición.

Feature-Driven Development (FDD)

- Nace de las experiencias de Jeff De Luca y Peter Coad en el desarrollo de una compleja aplicación comercial.

- Se centra en un proceso sencillo, basado en el modelado eficiente, e iteraciones cortas (2 semanas)
- Las practicas están dirigidas desde el punto de vista de las **features**, funcionalidades que aportan valor al cliente.
- El objetivo central es que el proceso de fondo sea un soporte más que una obligación.
- Se centra en las fases de diseño e implementación del sistema partiendo del listado de requisitos
  - los cuales ya deben estar capturados y comprendidos.

Dilbert...



© Scott Adams, Inc./Dist. by UFS, Inc.

Roadmap → Puntos Destacados

- Las metodologías ágiles surgen como respuesta a problemas reales
- Se basan en el sentido común, pero rompen con creencias arraigadas. Pero la metodología perfecta no existe
- Se están extendiendo con rapidez
- Características:
  - El desarrollo iterativo es bajo en riesgos mientras que el modelo en cascada es alto.
  - Descubrimiento y alivio de los riesgos de forma temprana
  - Acomoda y provoca el cambio al inicio del proyecto
  - Obligada comunicación y compromiso. Confianza y satisfacción desde el principio, éxito repetido.
  - Producto parcial desde el inicio. El producto final encaja mejor en los deseos del cliente
  - Seguimiento del progreso relevante, lo que conlleva una mejor predecibilidad. Mejora del proceso de forma temprana y regular
  - Más calidad, menos incidencias

Roadmap → Para Saber Más

- **Bibliografía**
  - **Agile & Iterative Development. A Manager's Guide**, de *Craig Larman.*
  - **Agile Software Development. Principles, Patters, and Practices** de *Robert C. Martin,*
  - **Balancing Agility and Discipline. A Guide for the Perplexed** de *Barry Bohem* y *Richard Turner*
- **Enlaces**
  - Metodologías Ágiles en MSDN : http://www.microsoft.com/spanish/msdn/arquitectura/roadmap_arq/heterodox.asp
  - Wikipedia: http://en.wikipedia.org/wiki/Agile_software_development

¿Preguntas...?