Introducción a las Metodologías Ágiles

La Metodología agiles surgen como una alternativa de desarrollo de sistemas con requerimientos cambiantes y donde se exige reducir drásticamente los tiempos de desarrollo manteniendo una alta calidad. Las Metodologías tradicionales en cambio han demostrado ser necesarias cuando los proyectos a desarrollar son de gran tamaño y donde comúnmente se necesita adoptar un proceso de desarrollo de software en el cual debido a que tiene que existir un estricto control del proceso conlleva a una rigurosa definición de roles, practicas y artefactos.

En Febrero del 2001 tras una reunión celebrada en Otawa, nace el término "ágil" aplicado al desarrollo de software. En esta reunión participa un grupo de 17 expertos de la industria del software, incluyendo a algunos de los creadores o impulsores de metodologías de software. Su objetivo fue esbozar los valores y principios que deberían permitir a los equipos desarrollar software rápidamente y respondiendo a los cambios que puedan surgir a lo luego del proyecto.
Se pretendía 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.
De la misma surgió el "Manifiesto ágil" que consta de valores y 12 principios.

Según el Manifiesto se valora:

- *Individuos e iteración en cambio de procesos y herramientas.*
  La gente es el principal factor de existo de un proyecto software. 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 este configure su propio entorno de desarrollo en base a sus necesidades.
- *Desarrollo de software en cambio de desarrollo de documentación,*
  La regla a seguir es "no producir documentos a menos que sean necesarios de forma inmediata para tomar un decisión importante". Estos documentos deben ser cortos y centrarse en lo fundamental.
- *Colaboración del cliente en cambio de un contrato de negociación.*
  Se propone que exista una interacción constante entre el cliente y el grupo de desarrollo. Esta colaboración entre ambos será la que marque la marcha del proyecta y asegure su éxito.
- *Responder al cambio en cambio de un plan fijo.*
  La habilidad de responder a los cambios que puedan surgir a lo largo del proyecto (cambios en los requisitos, en la tecnología, en el equipo, etc.) determina también el éxito o fracaso del mismo. Por lo tanto, la planificación no debe ser estricta sino flexible y abierta.

Principios de los Modelos Ágiles

1. *La mayor prioridad se basa en satisfacer al cliente entregándole software en forma temprana y continua de que le aporte un valor.*
2. *Los cambios de los requerimientos son bienvenidos Se capturan los cambios para que el cliente tenga una ventaja competitiva.*

3. *Entrega de software frecuente. Es conveniente entregar cada un par de semanas o de meses, con el menor tiempo posible entre entregas.*
4. *La gente del negocio y el grupo de desarrollo deben trabajar juntos durante todo el proyecto.*
5. *Desarrollo del proyecto mediante la motivación individual. Hay que brindarles a los individuos el entorno y el soporte que ellos necesitan para poder llevar a cabo el trabajo.*
6. *El método más conveniente y eficiente para transmitir información dentro del equipo de desarrollo es a través de una conversación cara a cara.*
7. *El software que funciona es la medida principal del progreso.*
8. *Los modelos ágiles promueven un desarrollo continuo.*
9. *Los sponsors, los desarrolladores y los usuarios deben tratar de mantener un buen clima durante todo el proyecto.*
10. *Siempre hay que tener en cuenta la simplicidad.*
11. *La mejor arquitectura, los mejores requerimientos y el mejor diseño emergen de la buena organización de los equipos.*
12. *En intervalos regulares, el grupo de desarrollo trabaja en como ser más efectivos y de esa manera realiza los ajustes que sean necesarios*[1].

Diferencia entre las metodologías agiles y tradicionales

La siguiente tabla presenta y resume las diferencias entre las metodologías ágiles y las metodologías tradicionales o estáticas, no solo en lo que refiere al proceso en si, si no también al contexto de equipo y organización que es más favorable a cada uno de estas filosofías de procesos de desarrollo de software.

1 – Manifiesto Agil http://agilemanifesto.org/

Metodologías Ágiles	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 preparados para cambios durante el proyecto	Cierta resistencia a los cambios
Impuestas internamente (por el equipo de desarrollo)	Impuestas externamente
Proceso menos controlado, con pocos principios	Proceso mucho más controlado, con numerosas políticas/normas
No existe 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	Más artefactos
Pocos roles	Más roles
Menos énfasis en la arquitectura del software	La arquitectura del software es esencial y se expresa mediante modelos

Metodologias Agiles en el Desarrollo de Software
Canós, Letelier y Penades
Universidad Politecnica de Valencia

Conveniencia de métodos ágiles

Aunque los métodos ágiles se diferencien en sus prácticas, ellos comparten un número de características comunes, incluyendo el desarrollo iterativo, y un enfoque sobre la interacción, la comunicación, y la reducción de artefactos intermedios caros en recursos. La conveniencia de los métodos ágiles en general, puede ser examinada desde múltiples perspectivas. Desde una perspectiva del producto, los métodos ágiles son más convenientes cuando los requerimientos son inesperados y cambian rápidamente; ellos son menos convenientes para los sistemas que tienen alta fiabilidad y exigencias de seguridad, y sistemas críticos, aunque no haya ningún acuerdo general completo sobre este punto. Desde una perspectiva de la organización, la conveniencia puede ser evaluada examinando tres dimensiones claves de una organización: cultura, gente, y comunicación. En relación con estas áreas un número de factores de éxito claves han sido identificados:

- *La cultura de la organización debe apoyar la negociación*
- *Se debe confiar en la gente*
- *Menos gente, pero más competente*

- *Las organizaciones deben aceptar las decisiones que los desarrolladores toman*
- *Las organizaciones tienen que tener un ambiente que facilite la comunicación rápida entre miembros del equipo*

El factor más importante es probablemente el tamaño del proyecto. A medida que el tamaño crece, la comunicación cara a cara se hace más difícil. Por lo tanto, los métodos ágiles son más convenientes para proyectos con pequeños equipos, con menos de 20 a 40 personas.[2]

Programación Extrema (XP)

Historia:

A mitad de 1980, Kent Beck y Ward Cunningham trabajaban juntos en la empresa Tektronix. Ellos encontraron la idea de las tarjetas CRC y patrones de diseño, mientras construían sistemas en Smalltalk. XP nació de esta colaboración.
Beck abrió su consultora privada, donde fue descubriendo varias de las practicas de XP, como, por ejemplo, trabajar en un mismo lugar físico.
A mitad de 1990, Beck fue contratado por Chrysler para ayudar en un nuevo proyecto basado en Smalltalk, para el cual, también, fue consultado Martín Fowler.
Kent Beck, considerado entonces el creador de XP, dice que esta metodología expresa lo que aprendió con y de Cunningham.

Es uno de los modelos ágiles que enfatiza la comunicación, simplicidad y retroalimentación.

Características de XP

En adición a las prácticas de los modelos ágiles para XP se adicionan:

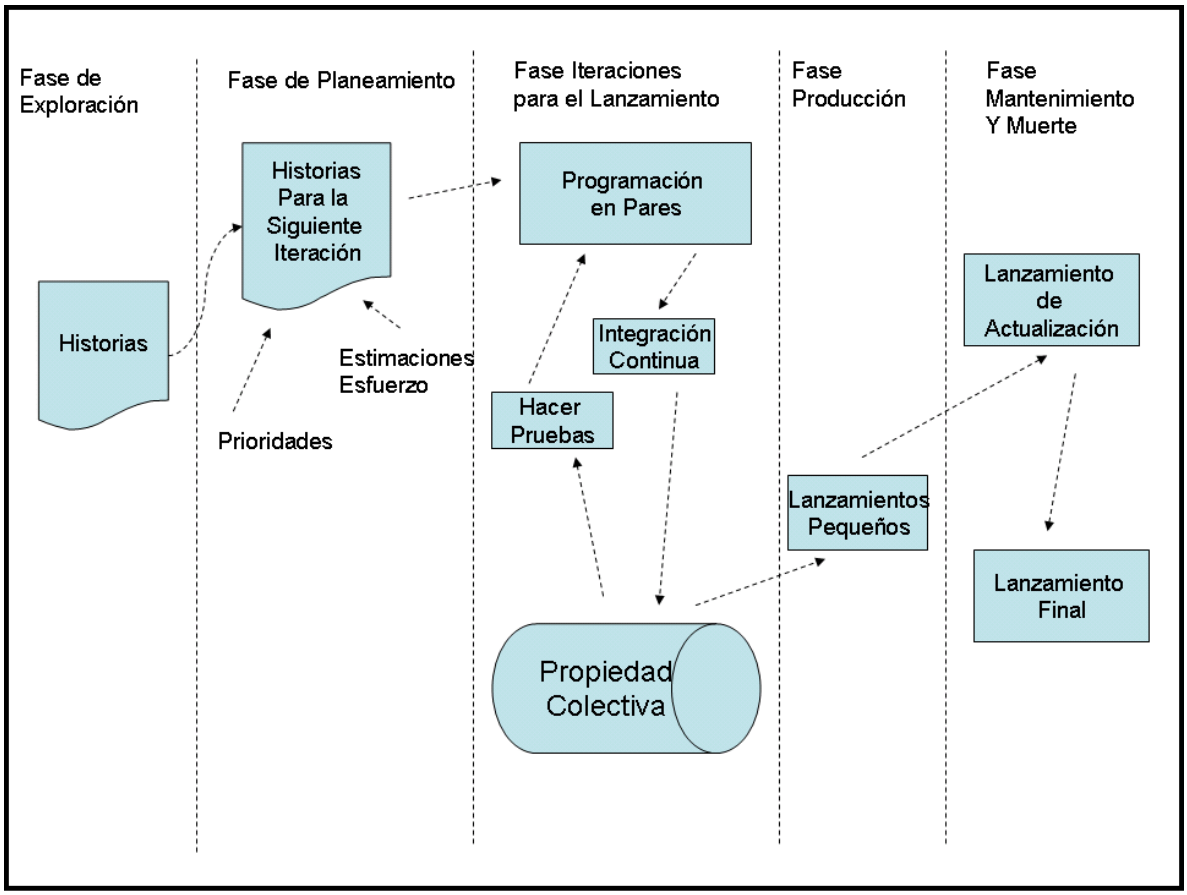
1. *Planear el juego*
2. *Entregas pequeñas y frecuentes.*
3. *Metáforas de sistema*
4. *Diseño simple*
5. *Testeo*
6. *Refactorización frecuente*
7. *Programación de a pares*
8. *Grupo de codificación*
9. *Integración continua*
10. *Armonía constante*
11. *Todo el grupo junto*
12. *Código standard*[3]

2 – Larman, Craig. 2004, Agile and Iterative Development: A manager´s guide. Addison Wesley
3 – Beck, Kent. 2002, Una explicación de programación extrema: Aceptar el cambio. Addison Wesley

Introducción

Es un método que se basa en la satisfacción rápida del cliente mediante entregas rápidas de alto valor para el mismo. Estas entregas deben ser continuas durante el desarrollo del proyecto. Se debe ser flexible al cambio. Esta enfocado para grupos de proyectos pequeños, con entregas menores a 1 año e iteraciones cortas, usualmente de una a tres semanas.
Como la palabra programación sugiere, este método provee ciertas reglas para los programadores; deben responder a los cambios de los requerimientos, aun avanzado el proyecto e igual, seguir produciendo código de calidad. Esto incluye desarrollo dirigido por el testeo, refactoreo, programación de a pares e integración continua.
XP esta orientado al equipo y a la comunicación, al cliente, a los desarrolladores y a los líderes del grupo de desarrollo. Se enfatiza que trabajen todos en el mismo lugar físico.
XP se distingue por no utilizar productos detallados, solo código y testeo.
Los métodos evolutivos realizan especificaciones de requerimientos y se realiza un plan para cada ciclo de entregas. Aparte se escriben detalles para la próxima iteración. En contraste, XP enfatiza la comunicación oral para el diseño y los requerimientos. Los requerimientos se escriben en las story card (tarjetas de historias). Los programadores descubren los detalles hablando con los clientes en el mismo lugar de trabajo.
La palabra extrema en XP viene de la convicción de Kent Beck que es trabajoso el desarrollo de software.

Ciclo de Vida



Exploración	Planeamiento	Iteraciones para la primer entrega	Producción	Mantenimiento
Propósito	**Propósito**	**Propósito**	**Propósito**	**Propósito**
Crear suficientes story cards para tener la información necesaria para encarar la primera iteración	Determinar la fecha de la primera entrega y en que va a consistir (que story cards va a incluir)	Implementar un sistema de testeo para la primera entrega	Desarrollo operativo	Construir la mayor parte de las entregas
Actividades	**Actividades**	**Actividades**	**Actividades**	**Actividades**
-Prototipos. -Exploración de las tecnologías de programación. -Escribir las story cards.	- Realizar un plan para la entrega. - Escribir las story cards.	-Testeo y programación. -Iteración del plan. -Escribir la tarea y realizar una estimación	Documentación Entrenamiento. Marketing.	Se debe incluir esta fase para la entrega incremental

Comentarios de Kent Beck acerca de las fases del ciclo de vida de XP:

.

1. *Como varios proyectos, XP puede empezar por exploración. Algunas historias de usuarios se escriben y se realizan estimaciones.*
2. *En la entrega del planeamiento, los clientes y los desarrolladores completan las historias de usuarios y las estimaciones, y deciden que realizar en la próxima entrega.*
3. *Para la próxima iteración (en el planeamiento de iteraciones) los clientes seleccionan cuales van a ser las que quiere implementar. Tienen que determinar las prioridades para la entrega. Los desarrolladores dividen las historias en tareas más pequeñas. Luego se hace una revisión del esfuerzo total estimado para las historias seleccionadas. XP no aconseja largas jornadas de trabajo, no más de 8 horas, para poder lograr un buen rendimiento por parte del grupo de desarrolladores. Se sostiene que si la gente no trabaja a gusto, baja la calidad y la productividad.*
4. *Los desarrolladores implementan las historias dentro del periodo determinado anteriormente, conjuntamente con los clientes que testean y proporcionan mas detalles sobre los requerimientos.*
5. *Si no se finaliza la entrega se deberá volver al punto 3) para la próxima iteración*

Roles en XP

I. Cliente

- *Escribe las historias y los tests de aceptación*
- *Selecciona las historias para release y para cada iteración.*

II. Desarrollador

1. *Programador*

- *Diseña, codifica y escribe los tests*
- *Refactorea*
- *Identifica tareas y realiza estimaciones*

2. *Tester*
   - *Ayuda al cliente a escribir y desarrollar los tests*

III. Management

1. *Coach*
   - *Enseña*

2. *Tracker*
   - *Recolecta métricas*
   - *Informa el progreso del proyecto*
   - *Realiza la retroalimentación de las estimaciones*

IV. Otros

- *Consultor técnico*

Practicas en las cuales se basa XP:

- *El grupo de desarrolladores y los clientes dentro del mismo lugar físico.*
- *El grupo de programadores y clientes trabajan en el lugar físico durante todo el tiempo de desarrollo. Los clientes tomaran decisiones sobre los requerimientos y sus prioridades.*
- *El cliente colabora en brindar detalles para escribir story cards y los tests de aceptación en colaboración con los programadores.*
- *En la primera entrega se necesita solo cliente que interactúe con el grupo de desarrolladores.*
- *Entregas pequeñas y frecuentes: entrega evolutiva. No es aplicable en todos los proyectos. No tiene que ser confundido con la organización de una entrega con un ciclo de varias iteraciones cortas.*
- *Testeo (testeo de aceptación y testeo del cliente): la practica de testeo dentro de XP es muy importante. Todos los puntos a desarrollar deben tener tests automáticos de aceptación. Todos los tests (aceptación y unidad) deben tener un resultado binario para que no sea necesario que haya una inspección humana cuando se necesite un resultado de un testeo. El test de aceptación es escrito en colaboración con el cliente; el define que entiende por aceptación. En XP esto es llamado test del cliente.*
- *Testeando: desarrollo dirigido por el testero y testeo de unidad. El testeo de unidad es escrito para la mayor parte del código y la practica del desarrollo guiado por el test continuo durante todo el proyecto. Esto incluye el testeo de la unidad que es escrito por el programador antes de escribir el código de lo que va a testear. Es un ciclo de test-código, en cambio de código-test. La aceptación y el test de unidad es corrido automáticamente y repetidamente en un ciclo continuo de integración y testeo.*

- *Planeamiento de la entrega: el planeamiento de la entrega tiene como propósito definir las expectativas de la próxima entrega operativa con el máximo valor software para el cliente. Típicamente en medio día de sesión, el o los clientes escriben story cards para describir los requerimientos y los desarrolladores realizan las estimaciones pertinentes. Existen story cards para la primera fase de trabajo. El cliente selecciona que realizar en la próxima iteración, teniendo en cuenta una fecha fijada y calcular cuantas tarjetas se pueden desarrollar durante el lapso determinado anteriormente o seleccionando tarjetas y luego calcular el día de la entrega.*
- *Planeamiento de la iteración: el planeamiento de la iteración consiste en seleccionar las tarjetas a implementar y elaborar un plan de tareas para desarrollarlas. El cliente selecciona las tarjetas para la iteración. Para cada una, los programadores crean una lista de tareas (en un par de tarjetas o en un borrador) para llevar a cabo lo descripto en las tarjetas. Luego se estimara cuanto tiempo se necesita para llevar a cabo la tarea. Si la tarea no esta estimada en medio día o en dos días, se refactorea.*
- *Diseño simple: el diseño debe contener un grupo mínimo de clases y métodos, y tiene que ser fácil de comprender.*
- *Programación de a pares: Todo el código es creado por dos programadores que utilizan la misma computadora. Los pares pueden cambiar frecuentemente para diferentes tareas. El observador hace una continua revisión del código y piensa más lentamente que el que tipea considerando los tests. Ciertamente, la productividad del grupo no depende simplemente de la cantidad de personas que tipean código sino de una combinación de prácticas en las cuales se basa XP.*
- *Refactorización frecuente: En XP se refactorea para simplificar el código y no hacer tan largo los elementos del diseño, sin cambiar la funcionalidad, mientras se pasan los tests. Se pretende un código simple y comprensible.*
- *Código es propietario del grupo: un par de programadores realizan un código, pero XP considera que el responsable es todo el equipo, no solamente los que hicieron el código. En XP, no existe el decir "él" o "ella" son responsables, sino que todo el equipo los es. Otra practica de XP es que cualquier persona del equipo puede modificar el código, no solo el par de programadores que lo realizan. Cuando se corre un test de aceptación y de integración es cuando se puede detectar un error en el código. Otro par de programadores lo modifican para poder ver el código desde otro punto de vista.*
- *Integración continua: todo código que es chequeado, es continuamente reintegrado y testeado en otra maquina separada; en un proceso automático de loop de compilación, test de unidad y test de aceptación.*
- *Armonía sostenida: es muy importante en XP mantener una armonía en el equipo para que todos trabajen contentos y produzcan más. Es importante que las personas se sientan cómodas para que el equipo rinda mas que la suma de sus integrantes.*
- *Código standard: Hay que tener un único estilo de programación, por que no solo se programa de a pares , sino también que cualquiera puede refactorear el código.*[3]

3 – Beck, Kent. 2002, Una explicación de programación extrema: Aceptar el cambio. Addison Wesley.

Productos de XP

- Story cards (Historias de usuario): Consiste en una tarjeta donde en forma manuscrita se escribe un requerimiento o un recordatorio para hablar con el cliente en detalle. Las historias de usuario de XP no son casos de uso ni escenarios. XP prefiere hablar con el cliente para que detalle los requerimientos que ser dirigidos por casos de uso como el Proceso unificado.
- Lista de tareas: durante el planeamiento de la iteración el equipo escribe en un pizarrón o en una hoja sobre la pared una lista de tareas para todas las historias de usuario seleccionadas para la iteración.
- Gráficos visibles: la idea es que sea sencillo para poder comunicarlo a todo el equipo. XP no realiza gráficos para documentación sino solamente para exponer y discutir ideas.

SCRUM

Historia

Las raíces de Scrum pueden ser encontradas en un artículo que resume las 10 mejores practicas de compañías japonesas. Este se denominaba "The New Product Development Game", Harvard Business Review Jan 1986, escrito por Takuchi y Nonaka.
El nombre de esta metodología, fue sacada del juego del rugby, por el cambio adaptativo del comportamiento del equipo durante el desarrollo del partido.
Uno de sus creadores es Jeff Sutherland, quien en la compañía Easel, en 1994 introdujo algunas de las practicas del articulo anteriormente mencionado e introdujo los encuentros diarios del equipo, que son el esqueleto principal en el cual se basa este método.
En 1996 Sutherland fundo Individual INC, y con el asesoramiento de Ken Schwaber redefinió y extendió las prácticas en las cuales se apoya Scrum.

Características

- ***Óptima para equipos de trabajo de hasta 8 personas**, aunque hay empresas que han utilizado Scrum con éxito con equipos más grandes, siempre que trabajen en el mismo lugar físico.*
- ***La documentación en Scrum no es indispensable**, como lo es en otras metodologías. Se puede comenzar el desarrollo de un proyecto aplicando Scrum sin documentar previamente.*
- ***El cliente va viendo mediante entregas parciales la evolución del producto**, cada 2 a 4 semanas.*
- ***Es indispensable que el cliente se involucre desde el comienzo en el proyecto.***
- ***Se hacen reuniones diarias de todos los equipos de desarrollo.***
- ***Diariamente se debe realizar un test del código producido e integrarlo con otro código existente.***

Roles

Los roles en Scrum son los siguientes:

- *Product Owner.*
- *Scrum Master.*
- *Scrum Team.*
- *Usuarios o Clientes.*

Product Owner: *Es el responsable oficial del proyecto. Conoce y marca las prioridades del mismo. Todos en el equipo deben respetar sus decisiones. Es quien debe dirigir y controlar el desarrollo del Product Backlog (una de las prácticas de Scrum).*

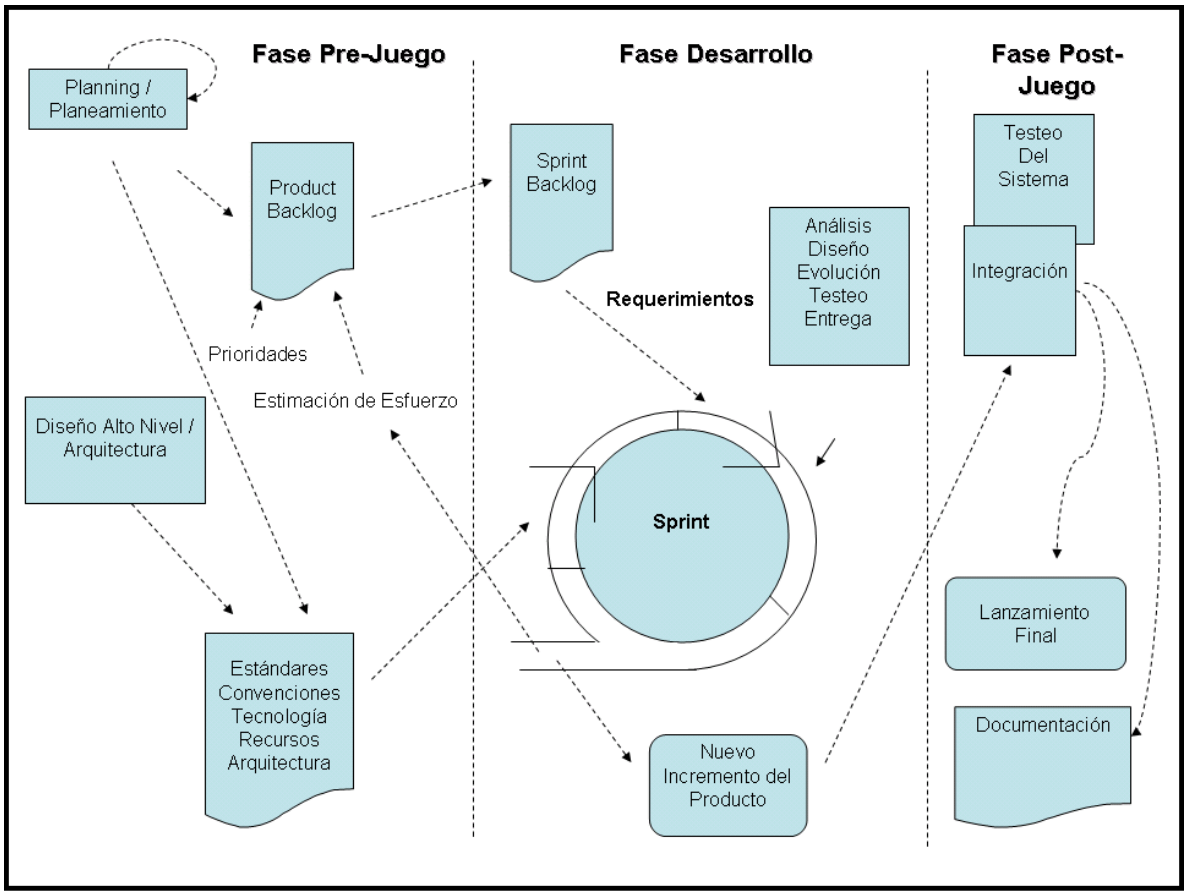
Scrum Master: *Es un nuevo rol de gestión introducido por Scrum. Es la persona encargada de asegurar que se cumplan las prácticas que establece Scrum, guiando las reuniones diarias, y viendo las dificultades que se le presentan a cada integrante del Scrum Team. También mide la evolución y las posibles desviaciones del objetivo de cada Sprint. También selecciona a las personas que conformarán el Scrum Team y controla las posibles presiones externas sobre este equipo.*

Scrum Team: *Es el grupo de personas encargadas de implementar la funcionalidad o funcionalidades elegidas por el* ***Product Owner.*** *El mismo debe estar conformado por analistas, diseñadores, expertos en control de calidad y programadores. Cada integrante de este equipo no trabaja de modo independiente, sino interactuando continuamente con los demás integrantes. No solo cumple su objetivo individual sino que colabora con las funciones de sus compañeros.*

Usuarios o Cliente: *Son los beneficiarios finales del producto, y deben colaborar con el equipo de desarrollo cuando este así lo establezca, aportando ideas, sugerencias o necesidades*[4].

4 – Schwaber, K, Beedle, M. 2002, Agile Software Development with Scrum. Prentice Hall

Actividades de Scrum



Las actividades fundamentales de Scrum son:

- **Product Backlog**
- **Sprint Planning Meeting**
- **Sprint Backlog**
- **Daily Scrum Meeting**
- **Sprint Review**
- **Sprint Retrospective**

Product Backlog: Consta de todas las tareas, funcionalidades o requerimientos a realizar. Por ejemplo permitir que los alumnos de una facultad puedan consultar sus materias aprobadas. Como vimos anteriormente es el **Product Owner** la persona que se encarga de marcar las prioridades, y mantener esta lista de tareas. Inicialmente esta lista es incompleta.

Sprint Planning Meeting: es una reunión en la que se seleccionan qué tareas del **Product Backlog** se realizarán en el siguiente **Sprint Backlog**. Participan de la misma **Usuarios o Clientes**, el **Product Owner**, **Scrum Master** y el **Scrum Team.** En este encuentro se define el **Sprint Goal,** o sea un documento o pequeña descripción de la siguiente funcionalidad o funcionalidades que se desarrollarán en el próximo **Sprint.** Es aquí también el **Product Owner** el encargado de priorizar las tareas.

Sprint Backlog: Se desarrollarán las tareas que fueron seleccionadas del **Product Backlog** en el **Sprint Planning Meeting.** Se recomienda que cada tarea sea llevada a cabo en un período de 2 a 4 semanas. Estas tareas no pueden ser modificadas una vez comenzado el actual **Sprint Backlog**, sino que formaría parte del próximo. Es durante la ejecución del **Sprint Backlog** donde se obtienen pequeñas piezas de software funcionando.

Daily Scrum Meeting: Durante cada **Sprint Backlog** se realiza una reunión diaria, con una duración de 30 minutos como máximo, con los integrantes del equipo, donde cada integrante del equipo debe informar:

- Qué ha hecho desde la última reunión.
- Qué tareas realizará en el día de la fecha.
- Con qué inconvenientes se ha enfrentado que impidan que realice la tarea asignada a él. El **Scrum Master** ayuda a resolver estos inconvenientes.

Sprint Review: El **Scrum** Team es el encargado de mostrar lo avances realizados en el Sprint, en una reunión que tiene una duración máxima de 2 horas. En la misma participan **Product Owner**, el **Scrum Master**, el **Scrum Team**, los **Clientes** o **Usuarios** y todos los stakeholders. En esta reunión los **Clientes** o **Usuarios** evaluarán las piezas de software ya desarrolladas.

Sprint Retrospective: Es el momento en que el **Product Owner** con el equipo evalúan si hay cambios o ajustes que deban ser realizados, como resultado de haber comparado el objetivo del **Sprint Backlog** con el resultado obtenido. Se observa qué aspectos positivos deberán reutilizarse y cuáles aspectos negativos tener presentes para no repetirlos.

Productos

Product Backlog: en el ejemplo que se muestra a continuación se pueden observar distintos puntos y su priorización hecha por el dueño del producto. Las estimaciones son realizadas en base a las horas-hombre necesarias para cada ítem.

REQUERIMIENTO	NUMERO	CATEGORÍA	ESTADO	PRIORIDAD	ESTIMACION
Proceso de pago en efectivo					
- PDA captura de venta	17	Caso de uso	sin empezar	4	60
- Proceso de pago con tarjeta de crédito	53	Tecnológica	sin empezar	1	100
- Calcular comisión sobre la venta	60	Caso de uso	sin empezar	5	30
		Caso de uso	completo	2	20

Sprint Backlog: En el ejemplo que se muestra a continuación, se muestran las estimaciones diarias para cada tarea. Una columna muestra la fecha y el total de horas.

DESCRIPCIÓN DE LA TAREA	RESPONSABLE	ESTADO	DÍA 6 7 8 9 10
- Encuentro para discutir los puntos a desarrollar	jms	completo	5 5 4 2 2
- Análisis de los datos	tn	en progreso	6 6 5 5 7
- Definición y construcción de la base de datos	gp	completo	12 12 12 12 12

No se permiten realizar gráficos de Pert: un grafico de Pert es construido asumiendo que todas las tareas del proyecto pueden ser identificadas, ordenadas, estimadas, que hay cambios mínimos y que se puede definir un proceso general. Esto es inconsistente con los principios de los modelos iterativos y ágiles.

En general se pueden utilizar algunos de los productos que utiliza el proceso UP para lo que se crea conveniente.

DSDM (Dynamic System Development Method)

Al igual que proceso unificado de rational, el DSDM es un framework que sirve para desarrollar un proceso de producción de software.
Es la combinación eficiente del conocimiento de las personas y técnicas para realizar proyectos rápidamente.
El equipo de desarrollo y usuarios trabajan juntos.
Se busca evitar producir sistemas que:

1. *No cumplan los requerimientos.*
2. *No funcionen correctamente*
3. *No caigan en desuso*

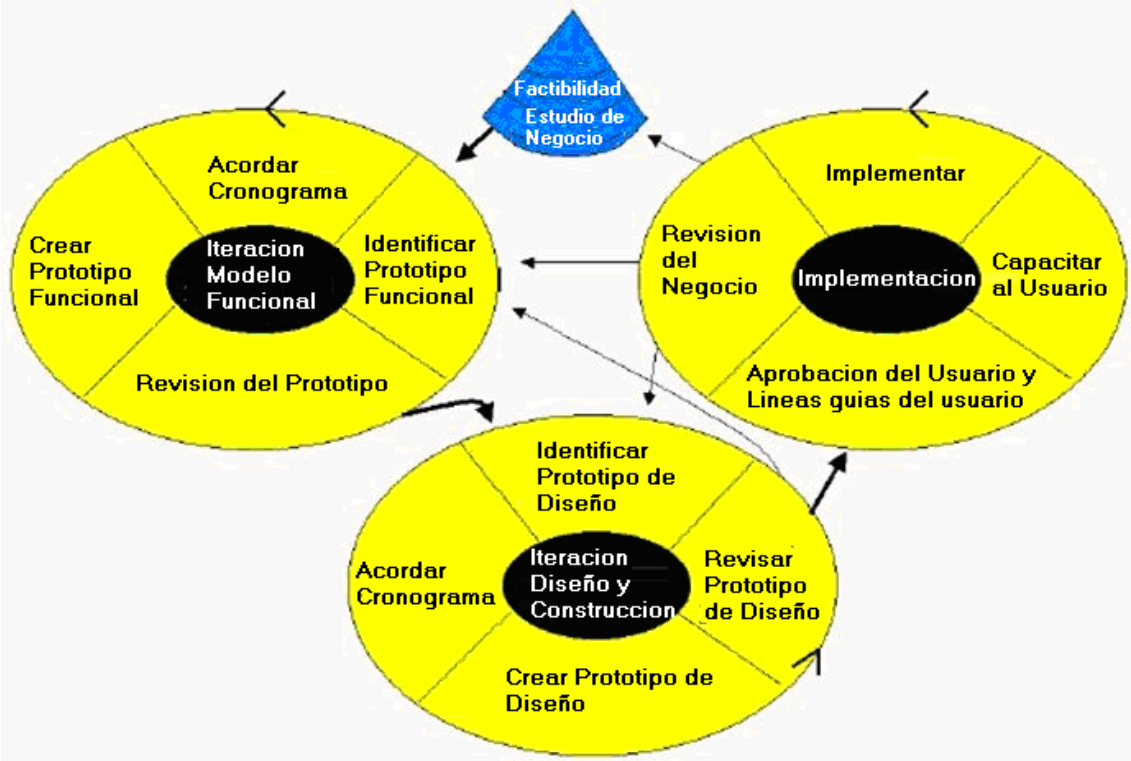
Es un proceso iterativo e incremental

Historia

El DSDM empezó en Gran Bretaña en 1994, como un consorcio de compañías del Reino Unido, que querían construir sobre RAD (Desarrollo Rápido de Aplicaciones) y desarrollo iterativo. Habiendo empezado con 17 fundadores ahora tiene más de mil miembros y ha crecido fuera de sus raíces británicas. Siendo desarrollado por un consorcio, tiene un sabor diferente a muchos de los otros métodos ágiles. Tiene una organización de tiempo completo que lo apoya con manuales, cursos de entrenamiento y programas de certificación.[5]

5- Tuffs, D., Stapleton, J., West, D., Eason Z. 1999, Interoperatibility of DSDM with the RUP. Rational and DSDM Consoltium.

Proceso



Proceso de desarrollo DSDM (www.dsdm.org)

DSDM consiste en cinco fases:

1. Estudio de viabilidad.
2. Estudio de negocio.
3. Modelo de Iteración Funcional.
4. Iteración de Diseño y Construcción.
5. Implementación

Las dos primeras fases son secuenciales y se realizan solo una vez. Las últimas tres fases se desarrollan en forma iterativa e incremental con iteraciones que son de tiempo fijo. El tiempo que insume cada iteración es planeada de antemano, de manera que el resultado de la iteración esta garantizado que se produzca. En DSDM, una duración típica para una iteración es desde unos pocos días hasta unas pocas semanas.
La fase de viabilidad es donde la conveniencia de DSDM para un proyecto dado es evaluada. Dependiendo del tipo de proyecto y del tipo de organizaciones y personas involucradas, se toma la decisión de utilizar o no el DSDM. En esta fase es donde se analizan las posibilidades técnicas y los riesgos. Se realizan dos productos: un reporte de viabilidad y un plan de desarrollo.
Opcionalmente, se puede realizar un prototipo rápido si el dominio del negocio o las tecnologías no son conocidos por el grupo de desarrollo.
Esta fase no tiene que durar más que un par de semanas.
En la fase de estudio del negocio es donde se analizan las características esenciales del negocio y la tecnología.

La recomendación es organizar workshops, con un número considerable de clientes expertos que puedan considerar todas las facetas relevantes del sistema y estar de acuerdo en establecer prioridades. Los procesos del negocio afectado y las clases de usuarios están descriptos en el área de definición del negocio. La identificación de las clases de usuarios afectados ayuda a involucrar al cliente, como la gente clave en la organización del cliente puede ser reconocida e involucrada en una etapa temprana del proyecto. Las descripciones de los procesos en un alto nivel están presentadas en la definición del área de negocio, en un útil formato (Diagramas de Requerimientos, Modelos de Objetos del Negocio, etc.).
Otras dos salidas son hechas en la fase de estudio del negocio. Una es la definición de la arquitectura del sistema y un plan del prototipo del contorno. La primera arquitectura del sistema puede cambiar a lo largo del desarrollo del proyecto DSDM.
El plan de prototipo va a contener la estrategia para cada etapa y el plan de gestión de configuración.
La fase de iteración del modelo funcional es la primera fase iterativa e incremental. En cada iteración, el contenido y el límite de cada iteración es planeado, mientras se lleva a cabo la iteración y el resultado es analizado en las futuras iteraciones.
Tanto el análisis como la codificación son realizadas en esta fase; los prototipos son construidos y no son descartados, sino que se le agregara funcionalidad para la construcción del sistema final.
La salida de esta fase es un modelo funcional que contiene un modelo de análisis y un prototipo de código y también un testeo continuo se incluye en ella.
Hay cuatro salidas en esta fase, en diferentes etapas de la misma. Una de ellas es la Lista de Prioridad de Funciones, que se va a ir entregando al final de cada iteración.
Los Documentos de Revisión del Prototipo, funcionan como un conjunto de comentarios del usuario. Esto sirve como una entrada para las iteraciones siguientes.
Lista de Requerimientos no Funcionales que serán considerados en la fase siguiente.
El Documento de los Riesgos de Análisis es importante en esta fase ya que en la fase de diseño pueden aparecer riesgos que sean más difíciles de abordar.
La fase iteración de diseño y construcción es donde el sistema es construido. La salida es el sistema testeado que satisface un mínimo conjunto de requerimientos. El diseño y la construcción iterativa y el diseño y prototipos funcionales son revisados por el usuario y el desarrollo esta basado en los comentarios de los usuarios.
La fase de implementación es donde el sistema es transferido del entorno del desarrollo al entorno de producción actual. Se debe entrenar a los usuarios al nuevo sistema. Si hay un gran número de usuarios de diferentes tipos se puede hacer más de una iteración de esta fase. Aparte de la entrega del sistema, la salida de la fase de implementación incluye el manual del usuario y un reporte de la revisión del proyecto.
DSDM define cuatro formas posibles de desarrollo.
Si el sistema satisface todos los requerimientos, no se necesita ningún otro trabajo extra.
Si una parte importante de los requerimientos se dejaron de lado porque no fueron descubiertos en una etapa temprana del proyecto, se debe volver a iniciar el proceso desde el comienzo hasta el final. Si algunos requerimientos funcionales de poca importancia fueron omitidos, hay que volver a iniciar el proceso desde la fase del modelo de iteración funcional. Por ultimo, si no se tuvo en cuenta algún requerimiento no funcional, se deberá iniciar el proceso desde la fase de iteración de diseño y construcción.

Roles y responsabilidades

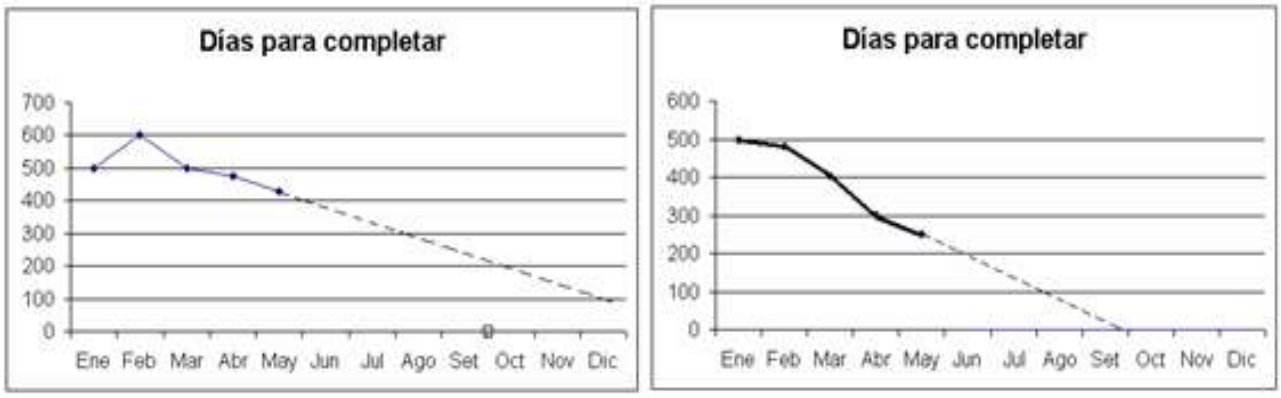
DSDM define 15 roles para los usuarios y los desarrolladores.
Desarrolladores y desarrolladores seniors son los únicos roles de desarrolladores. El desarrollador senior se basa en la experiencia de las tareas que lleva a cabo el desarrollador, e indica un titulo que es la jerarquía que posee dentro del equipo. Los dos roles cubren todo el grupo de desarrolladores, analistas, diseñadores, programadores y testeadores.
Coordinador técnico define la arquitectura del sistema y es responsable de la calidad técnica del proyecto. El también es responsable del control técnico del proyecto, como realizar la gestión de configuración del software.
De los roles del usuario el más importante es el Usuario embajador . Su tarea principal es brindar su conocimiento del negocio a los integrantes el grupo de desarrollo y distribuir la información acerca del avance del proyecto a los demás usuarios. Esto asegura una retroalimentación entre los usuarios y los integrantes del grupo de desarrollo. Este rol debe provenir de la comunidad del usuario que va a utilizar el sistema. Este no puede representar todos los puntos de vista de los diferentes usuarios, por lo que existe otro rol definido como Usuario avisador. Estos son usuarios que representan un punto de vista importante para el proyecto, como, por ejemplo, staff de tecnología de información, auditor financiero, etc.
El Visionario es un usuario participante que es el que posee la mejor percepción de los objetivos del negocio para el sistema y para el proyecto. El visionario es probablemente la persona del grupo de personas que tuvieron la idea inicial para construir el sistema. La tarea del visionario es asegurar que los requerimientos esenciales sean descubiertos y entendidos tempranamente por parte del grupo de desarrolladores, y que el proyecto vaya en la dirección correcta desde el punto de vista de dichos requerimientos.
El Sponsor ejecutivo es la persona de la organización del usuario que tiene autoridad financiera. Este el que tiene el poder para tomar decisiones.

Practicas:

Nueve prácticas definen la ideología y la base de todas las actividades de DSDM:

PRACTICAS DSDM	**Descripción**
La participación activa del usuario es fundamental.	Un grupo de usuarios deben estar presentes mientras se desarrolla el sistema para asegurar la retroalimentación en el tiempo adecuado
El equipo DSDM debe tener poder para tomar decisiones	Decisiones a largo plazo no son toleradas en un ciclo de desarrollo rápido. Los usuarios involucrados en el desarrollo tienen que tener el conocimiento para relatar que debe hacer el sistema.
El foco esta en la entrega frecuente de productos	Las decisiones erróneas pueden ser corregidas, si el ciclo de entregas es corto y los usuarios aseguran una buena retroalimentación.
La agilidad para el propósito del negocio es un criterio esencial para aceptación de los productos.	"Construir el producto correcto antes de construirlo de la mejor manera posible". No hay que postergar los requerimientos no funcionales para último momento.

El desarrollo iterativo e incremental es necesario para que el equipo de desarrollo se asegure llegar a la solución del negocio.	Los requerimientos principales del sistema permanecen intactos desde el comienzo hasta la finalización del proyecto. Pero si el sistema se desarrolla en forma iterativa, los errores pueden ser encontrados y corregidos tempranamente.
Todos los cambios durante el desarrollo, pueden ser revertidos.	En el transcurso del desarrollo, un paso erróneo puede ser cometido utilizando iteraciones cortas y asegurándose que se puede retornar a los estados previos, los pasos erróneos pueden ser corregidos.
Los requerimientos son definidos primeramente en un nivel alto.	El congelamiento de los requerimientos solo puede hacerse en un alto nivel, para permitir cambiar los detalles de los mismos cuando sea necesario. Esto asegura que los requerimientos esenciales son capturados en una etapa temprana, pero el desarrollo de los mismos puede comenzar antes que todos los requerimientos sean establecidos. Mientras que el desarrollo progresa, pueden descubrirse mas requerimientos a medida que se incrementa el conocimiento del sistema.
El testeo es integrado a través de todo el ciclo de vida.	Todos los componentes del sistema deben ser testeados por los desarrolladores y los usuarios a medida que son desarrollados. el testeo también es incremental.
La colaboración y la cooperación de todos los stakeholders es esencial	Para poder desarrollar con DSDM, la organización nizacion se debe comprometer a que todos colaboren con el proyecto, inclusive el departamento de información tecnológica. La elección de cada entrega del sistema y que se deja para el final es siempre un compromiso, y requiere un acuerdo en común. En menor escala, las responsabilidades del sistema deben ser compartidas por los usuarios y los desarrolladores.



Gráficos de quemado – Con necesidad de recortar retrasos (izq.) y con entrega proyectada en término.

Medición realizada en mayo – La fecha de entrega proyectada es el 1° de

Análisis comparativo de las metodologías ágiles

El objetivo es realizar una comparación de las metodologías ágiles descriptas anteriormente. Estas son:

- XP (Programación extrema)
- Scrum
- DSDM

Los criterios que se utilizaron para la comparación son:

- Características
- Ciclo de vida
- Roles
- Productos

Características de los modelos agiles

XP	SCRUM	DSDM
- Trabajar estrechamente con el cliente - Mini entregas, en tiempos reducidos (entre 2 y 4 semanas) - Pair Programming (programación de a pares). - 40 horas semanales de trabajo. - Proceso guiado por las pruebas (test driven development) - Propiedad colectiva del código. - Todos los integrantes del grupo de desarrollo y el cliente juntos en el mismo lugar físico. - Refactorización del código. - Integración frecuente.	- Equipos autodirigidos y autoorganizados. - una vez seleccionadas las tareas para una iteración, no hay adición externa de tareas a la misma. - Encuentros diarios con preguntas, previamente definidas. - Realizar una demo a los stakeholders externos al final de cada iteración. - Plan adaptativo dirigido por el cliente. - Todos los integrantes del grupo de desarrollo juntos en el mismo lugar físico.	- Entregas frecuentes. - Los cambios en los requerimientos son bienvenidos. - La implicación activa de los usuarios es imprescindible. - Desarrollo iterativo e incremental. - Todos los cambios durante el desarrollo pueden ser revertidos - La agilidad para el propósito del negocio es un criterio esencial para la aceptación de los productos. - El testeo es integrado a través de todo el ciclo de vida. - La colaboración y la cooperación de todos los stakeholders es esencial.

Roles

XP	SCRUM	DSDM
- Cliente - Desarrollador (programador y tester) - Management (coach, tracker) - Consultor tecnico	- Cliente - Desarrollador - Management - Chicken	- Desarrolladores seniors - Coordinador técnico - El usuario embajador - El sponsor ejecutivo - El visionario.

Ciclo de Vida

	Concepción del proyecto	**Especificación de los Requerimientos**	**Diseño**	**Codificación y Desarrollo**	**Testeo**	**Produccion**
XP	- Fase de exploración	- Fase de Exploración - Fase de planeamiento	- Fase de iteraciones para la primera entrega	- Fase de iteraciones para la primera entrega	- Fase de iteraciones para la primera entrega. - Fase de producción.	- Fase de mantenimiento
SCRUM	- Fase de planeamiento	- Fase de organización	- Fase de desarrollo	- Fase de desarrollo	- Fase de desarrollo	- Fase de release
DSDM	- Fase de estudio de viabilidad	- Fase de estudio del negocio - Fase de iteración del modelo funcional	- Fase de iteración de diseño y construcción	- Fase de iteración de diseño y construcción.	- Fase de iteración de diseño y construcción.	- Fase de implementación.

Bibliografía

Manifiesto Ágil http://agilemanifesto.org/

Agile and Iterative Development: A manager´s guide.
Larman, Craig.
Addison Wesley
2004

Una explicación de programación extrema: Aceptar el cambio.
Beck, Kent.
Addison Wesley
2002

Agile Software Development with Scrum.
Schwaber, K, Beedle, M.
Prentice Hall
2002

Interoperatibility of DSDM with the RUP.
Tuffs, D., Stapleton, J., West, D., Eason Z.
Rational and DSDM Consoltium.
2002