ASOCIACIÓN DE INVESTIGACIÓN
EN SOFTWARE INTELIGENTE

DISEÑO Y CONSTRUCCIÓN DE APLICACIONES ÁGILES CON EL MÉTODO SCRUM

METODOLOGÍAS ÁGILES Y PROGRAMACIÓN EXTREMA

Mayo 2007
La Paz - Bolivia

CONTENIDO

1
INTRODUCCIÓN

El desarrollo del software no es una tarea fácil, prueba de ello es que existen numerosas propuestas metodológicas que inciden en distintas dimensiones del proceso de desarrollo. Por una parte, existen aquellas propuestas tradicionales que se centran especialmente en el control del proceso, estableciendo rigurosamente las actividades involucradas, los artefactos que se deben producir, además de las herramientas y notaciones que se usarán. Estas propuestas han demostrado ser efectivas y necesarias en un gran número de proyectos, pero también han presentado problemas en otros muchos. Una posible mejora es incluir en los procesos de desarrollo más actividades, más artefactos y más restricciones, basándose en los puntos débiles detectados. Sin embargo, el resultado final sería un proceso de desarrollo más complejo que puede incluso limitar la propia habilidad del equipo para llevar a cabo el proyecto. Otra aproximación es centrarse en otras dimensiones, como por ejemplo el factor humano o el producto software. Esta es la filosofía de la metodología ágil, la cual da mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental del software con iteraciones muy cortas. Este enfoque está mostrando su efectividad en proyectos con requisitos muy cambiantes y cuando se exige reducir drásticamente los tiempos de desarrollo pero manteniendo una alta calidad. La metodología ágil está revolucionando la manera de producir software, genera un amplio debate entre sus seguidores y quienes por escepticismo o convencimiento no la ven como alternativa frente a las metodologías tradicionales [1].

1.1 HISTORIA

En febrero de 2001, tras una reunión celebrada en Utah - EEUU, 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 desarrollo del 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 largo del proyecto.

Se pretendía ofrecer una alternativa a los procesos de desarrollo del software tradicionales, caracterizados por ser rígidos y dirigidos por la documentación que se genera en cada una de las actividades desarrolladas [1].

La Tabla 1.1 recoge esquemáticamente las principales diferencias de la metodología ágil con respecto a las metodologías tradicionales (denominado en este contexto como “no ágil”). Estas diferencias que afectan no sólo al proceso en sí, sino también al contexto del equipo así como a su organización [1].

Tabla 1.1: Diferencias entre la metodología ágil y las tradicionales.

Metodología Ágil	Metodologías Tradicionales
Basada 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
Impuesta internamente (por el equipo)	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 de menos de 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.

1.2. MANIFIESTO ÁGIL

El manifiesto ágil es un resumen de los principios sobre los que se basan los métodos alternativos de desarrollo del software, y en su firma participaron: Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim

Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland y Dave Thomas, según este manifiesto ágil se valora [1]:

1. **Al individuo y las interacciones del equipo de desarrollo sobre el proceso y las herramientas.** La gente es el principal factor de éxito 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 éste configure su propio entorno de desarrollo con base en sus necesidades.
2. **Desarrollar software que funciona más que conseguir una buena 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.
3. **La colaboración con el cliente más que la negociación de un contrato**. Se propone que exista una interacción constante entre el cliente y el equipo de desarrollo. Esta colaboración entre ambos será la que marque la marcha del proyecto y asegure su éxito.
4. **Responder a los cambios más que seguir estrictamente un plan**. 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.

1.2.1. Principios ágiles

Los valores anteriores inspiran los 12 principios del manifiesto, estos principios son características que diferencian un proceso ágil de uno tradicional. Los dos primeros son generales y resumen gran parte del espíritu ágil. Los restantes tienen que ver con el proceso a seguir y con el equipo de desarrollo, en cuanto a las metas a seguir y la organización del mismo. Según [1], los 12 principios son:

1. La prioridad es satisfacer al cliente mediante tempranas y continuas entregas del software que le aporte un valor.
2. Dar la bienvenida a los cambios. Se capturan los cambios para que el cliente tenga una ventaja competitiva.
3. Entregar frecuentemente software que funcione desde un par de semanas a un par de meses, con el menor intervalo de tiempo posible entre entregas.

4. La gente del negocio y los desarrolladores deben trabajar juntos a lo largo del proyecto.
5. Construir el proyecto en torno a individuos motivados. Proporcionarles el entorno, el apoyo que necesitan y confiar en ellos para conseguir finalizar el trabajo.
6. El diálogo cara a cara es el método más eficiente para comunicar información al interior de un equipo de desarrollo.
7. El software que funciona es la medida principal de progreso.
8. Los procesos ágiles promueven un desarrollo sostenible. Los promotores, desarrolladores y usuarios deberían ser capaces de mantener una armonía de trabajo constante.
9. La atención continua a la calidad técnica y al buen diseño mejora la agilidad.
10. La simplicidad es esencial.
11. Las mejores arquitecturas, requisitos y diseños surgen de los equipos organizados por sí mismos.
12. En intervalos regulares, el equipo reflexiona respecto a cómo llegar a ser más efectivo, y según esto ajusta su comportamiento.

1.3 MÉTODOS ÁGILES

Los métodos ágiles actualmente constituyen una tendencia bastante aceptada para el desarrollo de productos software, debido a que estos se están convirtiendo en métodos populares entre los desarrolladores [2].

1.3.1 Métodos ágiles representativos

Aunque los creadores e impulsores de los métodos ágiles más populares han suscrito el manifiesto ágil y coinciden con los principios ya mencionados, cada uno de estos métodos tiene características propias y enfatizan en algunos aspectos específicos [1].

A continuación se resumen los métodos ágiles más importantes:

1. Métodos Cristal – Crystal Methods

Desarrollados por Alistair Cockburn, Crystal no se refiere a un método cerrado, sino a un conjunto de ellos, con los criterios para seleccionar y adecuar los más apropiados a cada proyecto [3,4,5].

Estos métodos están centrados en las personas del equipo de desarrollo que representa un factor clave, por lo que es necesario invertir esfuerzos en mejorar sus habilidades y destrezas, así como tener definidas políticas de trabajo en equipo. Estas políticas dependerán del tamaño del equipo. De esta manera se ha establecido una clasificación por colores, es decir, que Crystal identifica con colores diferentes cada método y su elección debe ser consecuencia de los parámetros de critica así como del tamaño del sistema, que también involucra a la cantidad de miembros en el equipo de desarrollo [3,4,5].

En la figura 1.1, se observan los métodos cristal en función de los factores de criticidad y tamaño.

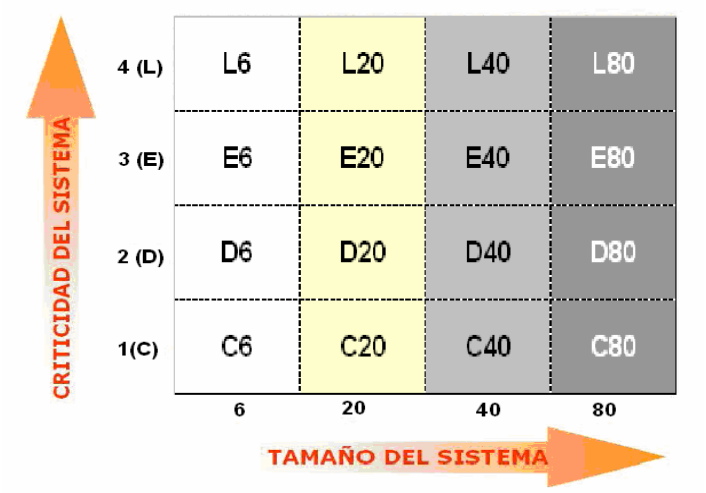

Figura 1.1: Métodos cristal en función a la criticidad y tamaño.
Palacio, 1.1

Las características de criticidad y dimensión de este método, figura 1.1, son descritas a continuación [3, 5]:

a. Criticidad, es la medida cuantitativa del numero de pérdidas que un mal funcionamiento del sistema ocasionaría al desarrollo del mismo. De abajo hacia arriba, los criterios que corresponden a los niveles de integridad y estos son:
   - Pérdida de confort o usabilidad.
   - Pérdidas económicas moderadas.
   - Pérdidas económicas graves.
   - Pérdida de vidas humanas.

b. Dimensión, determina el tamaño del sistema por el número de personas empleadas en su desarrollo, Cuantas más personas estén implicadas, más grande será el proyecto, y en este caso un error no detectado puede ser crítico además que implica un costo considerable. De izquierda a derecha, la dimensión está descrita por el número de miembros del equipo desde el más pequeño de 6 personas hasta el más grande de 80 personas, diferenciándose así la dimensión a través de colores.

Es de esta manera que la familia de métodos Crystal selecciona al más adecuado para el proyecto a realizar. La familia Crystal se fundamenta en [3,5]:

a. Desarrollo iterativo e incremental
b. Duración máxima de una iteración de 1 a 3 meses.
c. Énfasis en la importancia de las personas que en los procesos.
d. Énfasis en la comunicación directa.
e. Es un modelo abierto a la adaptación e introducción de prácticas de otros métodos ágiles, tales como Programación Extrema, Scrum, etc.

2. Desarrollo veloz en internet - Internet Speed Development (ISD)

El desarrollo veloz en Internet (Internet Speed Development (ISD)), surge de las conclusiones del coloquio celebrado en Octubre de 2001, que reunió a especialistas de las universidades Carneige Mellon, Georgia y Copenhagen. Sienta bases de gestión para abordar el desarrollo de sistemas de software, de tamaño pequeño que requieren tiempos de desarrollo muy rápidos [3].

3. Desarrollo de software adaptativo - Adaptative Software Development (ASD)

El Desarrollo de Software Adaptativo (Adaptative Software Development (ASD)) tiene su impulsor en Jim Highsmith, es un modelo de implementación de patrones ágiles para el desarrollo de software [3,5].

El ciclo de vida que propone tiene tres fases esenciales [3,5]:

a. Especulación, fase en la que se inicia el proyecto y se planifican las características del software, esta fase está compuesta por 5 pasos:
    i. Inicio para determinar la misión del proyecto.
    ii. Determinación del marco temporal del proyecto.
    iii. Determinación del número de iteraciones y la duración de cada una.
    iv. Determinación del objetivo de cada iteración.

v. Asignación de funcionalidad a cada iteración.

b. Colaboración, fase en la que se realiza un desarrollo concurrente de la construcción y gestión del producto

c. Aprendizaje, fase en la que:

i. Se revisa su calidad, con criterios del cliente y técnicos. La revisión de los componentes sirve para aprender de los errores y volver a iniciar el ciclo de desarrollo.

ii. Se entrega al cliente el producto

Sus principales características son:

a. Iterativo,

b. Orientado a los componentes software más que a las tareas y

c. Tolerante a los cambios.

4. Método para el desarrollo de Sistemas Dinámicos

Desarrollado y concebido por el denominado Consorcio DSDM (Dynamic System Development Method Consortium). DSDM es un modelo de procesos para el desarrollo de software, el cual estuvo representado en la firma del Manifiesto Ágil, el año 2001 la versión 4.1 fue considerada un método ágil [3,5].

El proceso del ciclo de desarrollo está compuesto de 5 fases, precedidas de un pre-proyecto y un post-proyecto, estas fases son [3,5]:

Fase 1. Estudio de viabilidad.

Fase 2. Estudio de negocio.

Fase 3. Iteración de modelado funcional.

Fase 4. Iteración de diseño y desarrollo.

Fase 5. Implementación.

Las tres últimas son iterativas, además de existir realimentación a todas las fases.

En la figura 1.2, se observa el diagrama de proceso DSDM y se describe de manera resumida los pasos que se realiza en dicho proceso.

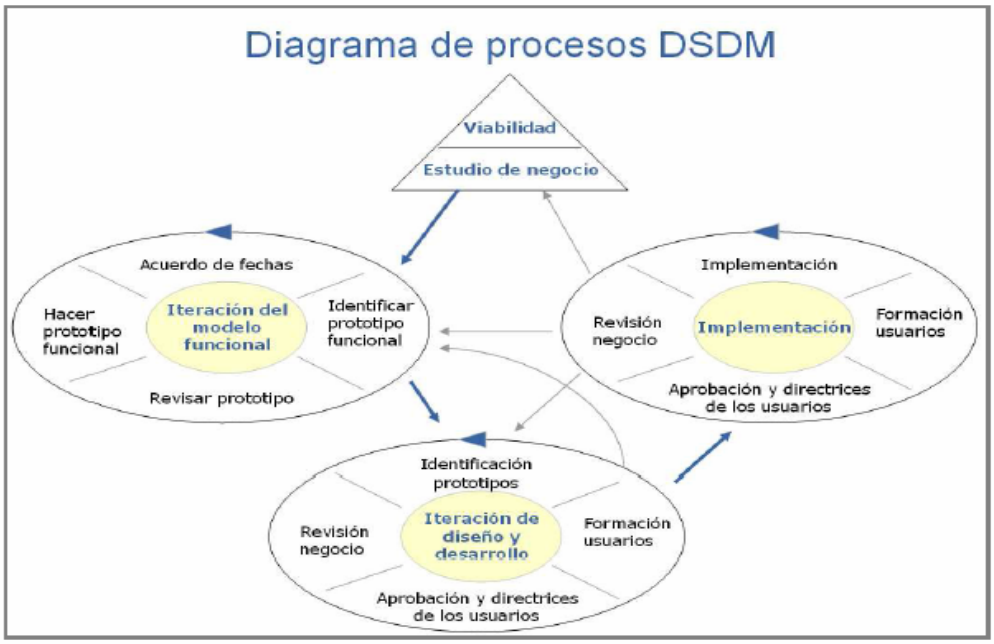

Figura 1.2: Diagrama de procesos DSDM

Palacio, 1.2

5. Desarrollo Guiado por Rasgos

Sus impulsores son Jeff De Luca y Peter Coad. El Desarrollo Guiado por Rasgos (Feature Driven Development (FDD)) tiene un proceso iterativo de 5 pasos, con iteraciones cortas de hasta dos semanas. Centrado en las fases de diseño e implementación del sistema partiendo de una lista de características que se desea reúna el software [3,5].

6. Desarrollo Continuo y Desarrollo Continuo del Software

El Desarrollo Continuo y Desarrollo Continuo del Software (Lean Development (LD) y Lean Software Development (LSD)) definida por Bob Charette, tiene como base la experiencia en proyectos con la industria japonesa del automóvil en los años 1980. Los cambios se consideran riesgos, pero si se manejan adecuadamente se pueden convertir en oportunidades que mejoren la productividad del cliente. Su principal característica es introducir un mecanismo para implementar dichos cambios [3,5].

Otros aspectos esenciales del LSD son la relación participativa con el empleado y el trato que le brinda la compañía, así como una especificación de principios, disciplinas y métodos iterativos, adaptativos, auto-organizativos e interdependientes en un patrón de ciclos de corta duración

que tiene algo más que un aire de familia con el patrón de procesos. LSD se inspira en doce valores centrados en estrategias de gestión [5]:

a. Satisfacer al cliente es la máxima prioridad.
b. Proporcionar siempre el mejor valor por la inversión.
c. El éxito depende de la activa participación del cliente.
d. Cada proyecto es un esfuerzo de equipo.
e. Todo se puede cambiar.
f. Soluciones de dominio, no puntos.
g. Completar, no construir.
h. Una solución al 80% hoy, en vez de una al 100% mañana.
i. La simplicidad es esencial.
j. La necesidad determina la tecnología.
k. El crecimiento del producto es el incremento de sus prestaciones, no de su tamaño.
l. Nunca empujes más allá de los límites del proyecto.

7. Programación Extrema

La Programación Extrema (Extreme Programming (XP)) es un método ágil centrado en potenciar las relaciones interpersonales como clave para el éxito en 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, la comunicación fluida entre todos los participantes, la simplicidad en las soluciones implementadas y el coraje para enfrentar los cambios, es especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico [3,6,7].

2

SCRUM

El grupo Standish “The CHAOS Report” el año 1994, informó que el 84% de los proyectos software de ese año habían fracasado, es decir, que el 32% de ellos se cancelaron, esto es alrededor de unos 80.000 proyectos, y 52% se quedaron fuera de presupuesto ya que sobrepasaron el costo inicial estimado en más del 189% [8,9].

Solamente el 16% de los proyectos concluyeron de manera exitosa. Además en términos de tiempo y presupuesto, estos tienen un 42% de la funcionalidad que se había planificado originalmente. Este tipo de problemas se atribuyen a la utilización de métodos pesados, como por ejemplo el Proceso Unificado de Desarrollo (RUP), Técnica de Modelado de Objetos (OMT) y otros usados en el desarrollo de productos software. Scrum, en el área de ingeniería de software, es un método ágil para el desarrollo de proyectos. Toma su nombre así como los principios, de los estudios realizados por Hirotaka Takeuchi e Ikujijo Nonaka sobre nuevas prácticas de producción a mediados de 1980, quienes se basaron en el juego de Rugby Scrum que es un deporte muy difundido en países anglosajones en el que se enfrentan dos equipos dentro de un campo rectangular en cuyos extremos se encuentran instaladas dos porterías con forma de "H", las características principales de este juego son el trabajo en equipo el cual se maneja por sí mismo y la adaptación [10,11].

Scrum surgió como modelo para el desarrollo de productos tecnológicos, pero también se emplea en entornos que trabajan con requisitos inestables y que además requieren rapidez y flexibilidad; estas situaciones son frecuentes en el desarrollo de determinados sistemas software [10,11].

En 1993 Jeff Sutherland identificó paralelismos con la industria del software, aplicó un modelo de desarrollo ágil, iterativo e incremental para desarrollar y mantener sistemas de software en Easel Corporation (Empresa que en los macro-juegos de compras y fusiones se integraría en VMARK, luego en Informix y finalmente en Ascential Software Corporation) al cual lo denomino Scrum. En 1996 presentó “Scrum” junto con Ken Schwaber como proceso formal, también para gestión de proyectos software en la Conferencia del año 1996, sobre Lenguajes, Sistemas y Programas Orientados a Objetos (OOPSLA 96). Ambos concibieron Scrum a partir de estudios científicos sobre procesos dinámicos [5,8,11]. El año 2001 fueron dos de los promulgadores del Manifiesto Ágil [5,11].

Scrum es un método que enfatiza un grupo de valores y prácticas para el proyecto y su continuación, más que los requerimientos, implementación u otros, lo empírico más que los procesos definidos. Además de ser fácilmente combinable con otros métodos [11].

En la figura 2.1, se observan los elementos de un proyecto Scrum ubicados en el proceso de desarrollo.

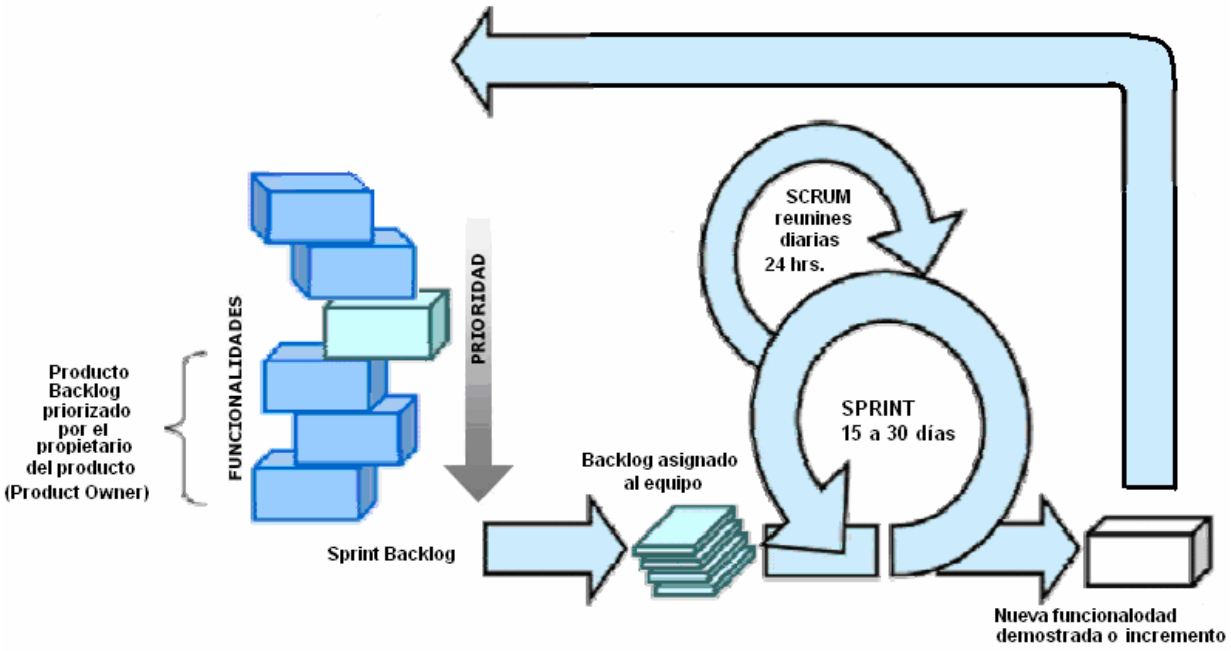

Figura 2.1: Visión general del proceso
Palacio, 2.1

2.1 ELEMENTOS DE UN PROYECTO SCRUM

Esencialmente se puede identificar 3 elementos principales que se utilizan durante el proyecto: Pila del producto, Pila del Sprint y el Incremento, los cuales se observan en la figura 2.2.

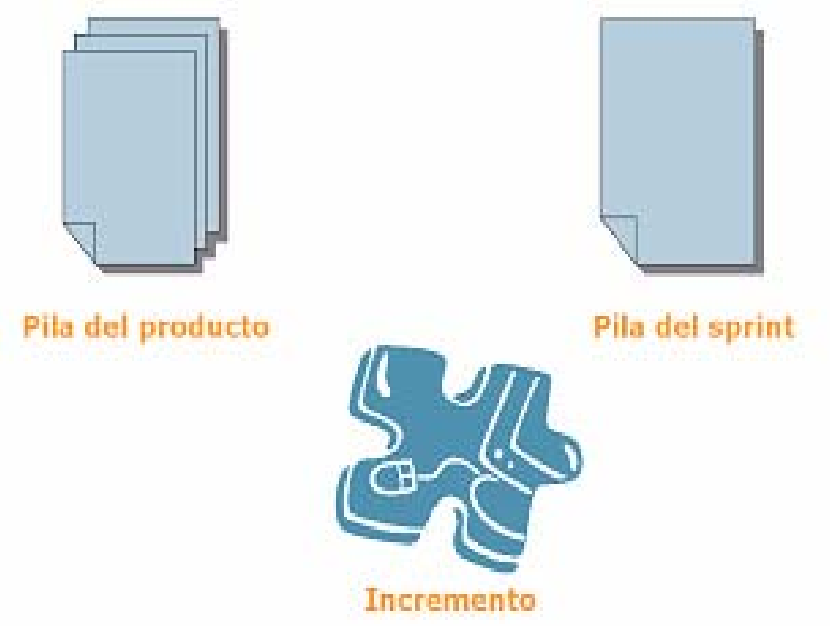

Figura 2.2: Elementos que se utilizan en SCRUM

Palacio, 2.2

A continuación se describen estos elementos [12,13]:

1. Pila del producto (Backlog del Producto)

Es una lista de requisitos de usuario que se origina con la visión inicial del producto y va creciendo y evolucionando durante el desarrollo; de acuerdo con [12], las características principales de esta pila son:

a. Priorización de acuerdo a la importancia que le da el propietario del producto.
b. Debe ser accesible para todos los miembros del equipo del proyecto.
c. Todos pueden contribuir y aportar elementos.
d. El responsable directo es el propietario del producto.

En la figura 2.3, se observa un ejemplo del backlog del producto.

Backlog del Producto

Id	Prioridad	Módulo	Descripción	Est.	Por
1	Muy alta		Plataforma tecnológica	30	JM
2	Muy alta		Prototipos interfaz usuario	40	LR
3	Muy alta		Diseño de datos	40	LR
4	Alta	Trastienda	El operador define el flujo y textos de un expediente	60	JM
5	Alta	Trastienda	Selección del lenguaje de programación.	60	JM
...					
20	Alta	Inventario	Mantenimiento.	999.	XX

Figura 2.3: Ejemplo del backlog del Producto

Palacio, 2.3

2. Pila del Sprint (Backlog del Sprint)

Se define “Sprint” como una iteración que dura alrededor de 30 días. La Pila del Sprint es una lista de los trabajos que debe realizar el equipo durante esta iteración para generar el incremento previsto, cuyas características principales son [12]:

a. Debe contener las funcionalidades que se van a realizar durante el Sprint.
b. El equipo debe estar comprometido a realizar dichas funcionalidades.
c. Se debe asignar tareas a los distintos miembros del equipo del proyecto.
d. Se debe hacer una estimación de cada funcionalidad.

En la figura 2.4, se observa un ejemplo del backlog del Sprint.

SPRINT	INICIO	DURACIÓN
1	01-feb-06	20

	X	J	V	L	M	X	J
	01-feb	02-feb	03-feb	06-feb	07-feb	08-feb	09-feb
Tareas pendientes	12	12	11	9	8	6	5
Horas de trabajo pendientes	176	164	144	111	93	76	50

PILA DEL SPRINT

Prod. ID	Tarea	Tipo	Estado	Responsable							
1	Tecnología web:dhtml, ajax, jscript?	Análisis	Terminada	Luis	24	20	14				
1	Tecnología reg. Datos	Análisis	Terminada	Luis	8	8	8				
2	Protot. Interfaz cliente web	Prototipado	Terminada	Luis	16	16	16	10	5		
2	Protot. Interfaz trastienda.	Prototipado	Terminada	Elena	16	8					
3	Diseño esquema XML.	Análisis	Terminada	Elena	16	16	10	5			
3	Diseño Base de datos	Análisis	Terminada	Elena	16	16	16	16	8		
4	Marco de Interfaz y menú	Codificación	Terminada	Luis	8	8	8	8	8	8	
4	Editor de flujograma.	Codificación	En curso	Elena	24	24	24	24	24	20	10
4	Editor de texto	Codificación	En curso	Luis	24	24	24	24	24	24	16
4	Editor XML	Codificación	En curso	Luis	8	8	8	8	8	8	8
...	...	...		...	...	...	...	...	...	...	...
20	Pruebas de edición de flujograma.	Pruebas	Pendiente	Elena	8	8	8	8	8	8	8
21	Pruebas de edición de expedientes.	Pruebas	Pendiente	Elena	8	8	8	8	8	8	8

Figura 2.4: Ejemplo de una Pila del Sprint

Palacio 2.4

También ayuda de gran manera una representación gráfica, ver figura 2.5, sobre las horas de tareas faltantes estimadas en el backlog del Sprint.

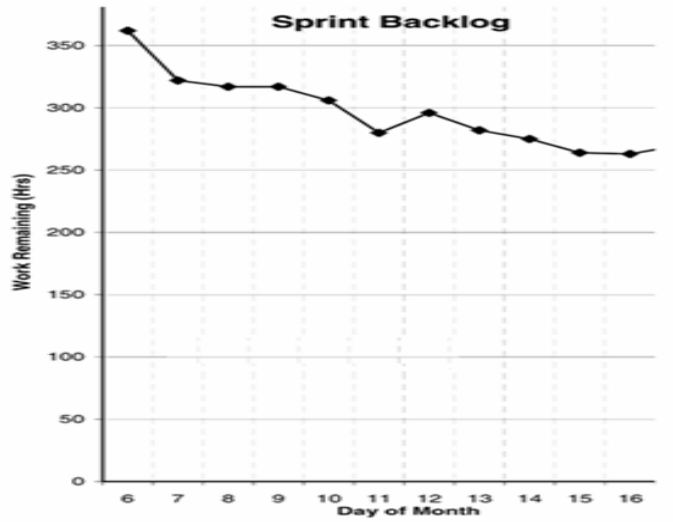

Figura 2.5: Gráfica del Backlog del Sprint
Larman, 2.5

3. Incremento

Es el resultado de cada Sprint, cuyas características principales son:

a. Es parte del producto desarrollado en un Sprint.
b. Debe estar en condiciones de ser usado.
c. Es una funcionalidad.

2.2 MODELO DE PROCESOS

Scrum es un método de desarrollo muy simple, que requiere mayor trabajo, ya que no se basa en el seguimiento de un plan, sino en la adaptación continua a las circunstancias de la evolución del proyecto [12,14].

Scrum es un método ágil, y como tal [12,14]:

1. Es una forma de desarrollo de carácter adaptable más que predictivo.
2. Está orientado a las personas más que a los procesos.
3. Emplea la estructura de desarrollo ágil: incremental basada en iteraciones y revisiones [5,12,14].

En la figura 2.6, se observa la manera en la que estructura el desarrollo ágil de un proyecto.

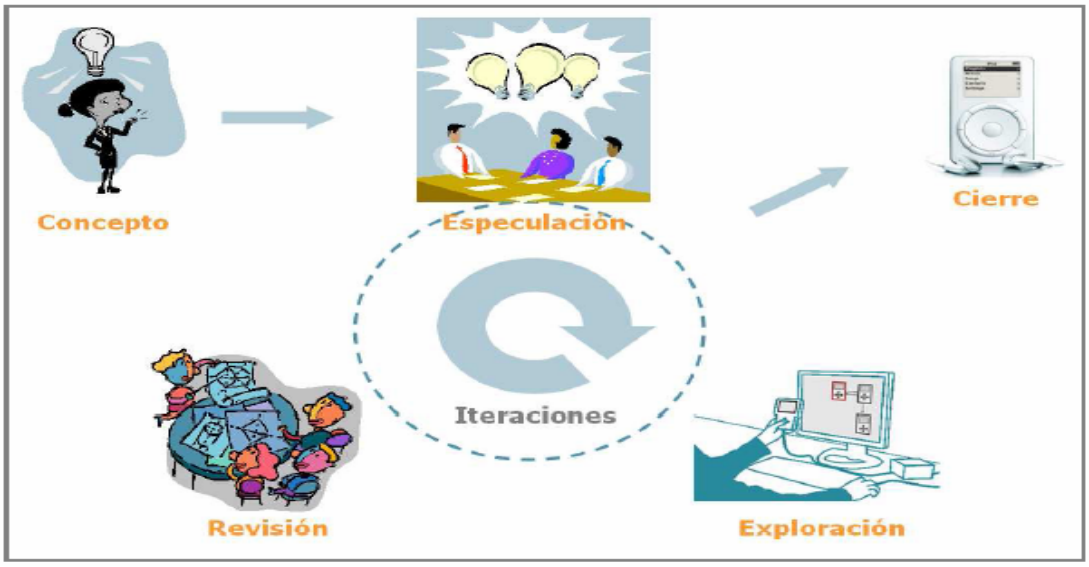

Figura 2.6: Estructura del desarrollo ágil
Palacio, 2.6

El modelo de proceso empieza con una visión general del producto, enfatizando en el detalle y en la especificación de las funcionalidades que tienen mayor prioridad de desarrollo y que pueden llevarse a cabo en un periodo de tiempo breve, por lo general 30 días [5,9,12].

Cada uno los períodos de desarrollo es una iteración que finaliza con la producción de un incremento operativo del producto. Estas iteraciones son la base del desarrollo ágil.

Scrum gestiona su evolución a través de reuniones breves diarias en las que todo el equipo revisa el trabajo realizado el día anterior y el previsto para el día siguiente [1,5,9].

En la figura 2.7, se observa la manera en la que se desarrollan los backlogs del producto y del Sprint, que constituyen la estructura central de Scrum.

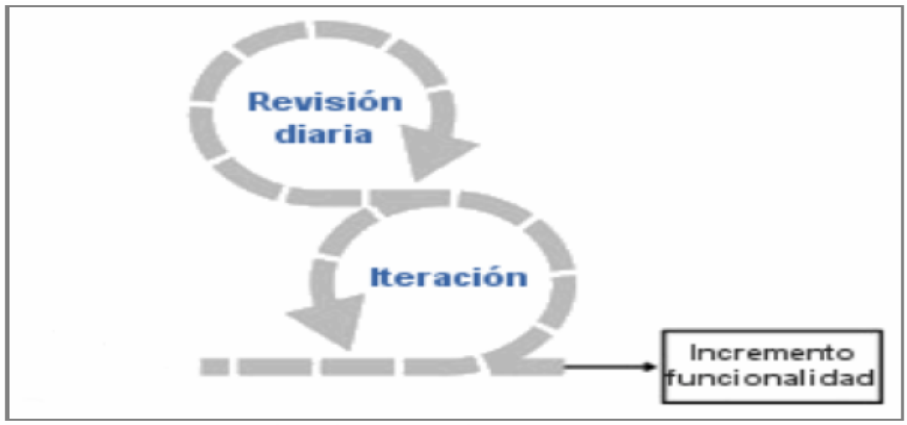

Figura 2.7 Estructura central de Scrum
Palacio, 2.7

La gerencia de los proyectos Scrum considera los siguientes aspectos [9]:

1. Desarrollo de software en etapas incrementales.
2. Requiere entregas de software terminado.
3. Se enriquece con la experiencia del equipo.
4. Se enfoca y potencia el trabajo en equipo.
5. Es fácil de aprender además de adaptable.
6. Adopta funcionalidad fácilmente.
7. La implementación de Scrum es de bajo riesgo y costo.

Scrum ya ha sido utilizado en más de 1.000 proyectos con éxito, entre ellos: Yahoo, Niké, PayPal, Microsoft, etc. Se aplica en áreas financieras, gobierno, productos comerciales, biotecnología, tiempo real, Internet y comercio electrónico. Scrum define métodos de gestión y control para complementar la aplicación de otros métodos ágiles como XP que, centrados en prácticas de tipo técnico, carecen de ellas.

2.3 VENTAJAS

Es una opción útil para el desarrollo de proyectos software por las siguientes razones [15,16]:

1. Evita estancamientos en el proyecto, existe un control del avance continuo por lo que cualquier error, falla o duda es resuelta antes de que esta se convierta en un problema mayor y lleve al equipo a parar el proyecto.
2. Seguimiento del proyecto, por medio de reuniones cortas y reuniones largas de planificación.

3. Seguimiento del equipo, se refleja también en las reuniones ya que si algún miembro del equipo tiene dificultades, existe un apoyo mutuo, Scrum hace énfasis en la importancia del equipo de desarrollo.
4. Software que incrementa su funcionalidad en cada sprint, ya que se hace una revisión de los entregables, a los cuales se les puede aumentar funcionalidades en el siguiente Sprint.
5. Mecanismos de control para variables cambiantes con el entorno.
6. Progreso en el producto con requerimientos inestables, como en los demás métodos ágiles, los requerimientos se hacen cada vez más específicos a medida que el proyecto avanza.
7. Aumenta comunicación con el equipo, debido al seguimiento continuo a través de las reuniones, aún cuando estas no se den en el mismo ambiente de trabajo.
8. El cliente obtiene retroalimentación frecuente sobre el producto, se mantiene al cliente constantemente informado sobre el avance del proyecto de tal manera que es él quien va evaluando que el producto funcione como él requiere que lo haga.
9. Se adapta fácilmente a otros métodos.

2.4 INCONVENIENTES

A continuación se listan los principales inconvenientes de Scrum [15,16]:

1. Existe una mínima supervisión. Esto quiere decir que Scrum hace énfasis en el ciclo de vida y los aspectos del manejo de desarrollo del proyecto más que en las técnicas de ingeniería de requerimientos por ejemplo.
2. El propietario del producto puede tener una visión de sus requerimientos, pero los describe rápidamente en el backlog del producto. Scrum no define como plasmar este ni otros aspectos en los productos de trabajo.
3. Los proyectos necesitan al menos un poco de documentación, Scrum no define cuál podría ser. Es así que no existe un estándar en cuanto al producto del trabajo, que pueda ser compartido con nombres en común a lo largo del proyecto, lo cual impide reusabilidad de los productos de trabajo y un vocabulario en común en general para las organizaciones.
4. Scrum, durante el desarrollo y la etapa de pruebas, no es descriptivo.

2.5 PRINCIPIOS

Puede considerarse como el principio, y de mayor importancia el de las reuniones diarias o reuniones de Scrum, la cual es descrita en detalle a continuación [13,16].

1. Reuniones de Scrum

La reunión de Scrum o simplemente llamada Scrum, constituye el núcleo del método y del proyecto en desarrollo. Esta reunión se trata de que cada día laboral, a la misma hora y en el mismo lugar, se lleve a cabo reuniones con los miembros del equipo de desarrollo.

Debido a que la reunión es sumamente corta, ya que no debe llevar más de 15 minutos, se la realiza en círculo y de pie. En este tiempo son contestadas por cada miembro del equipo de desarrollo las mismas preguntas especiales cada día [13,16]:

a. ¿Qué hiciste desde el último Scrum?
b. ¿Qué tareas realizarás hasta el siguiente Scrum?
c. ¿Qué es lo que te está impidiendo alcanzar el objetivo de la iteración?

Existen además otras dos preguntas, que son de mucha utilidad:

d. ¿Existe alguna tarea que haya sido olvidada y que debe añadirse al backlog del Sprint?, la cual no se refiera a nuevos requerimientos
e. ¿Has aprendido algo nuevo de los aspectos relevantes de los miembros del equipo?, esta pregunta da paso a un foro de aprendizaje de grupo continuo, la cual es vital para un desarrollo ágil.

Entre otros puntos se deben considerar [13,16]:

a. Deben durar de 15 a 20 minutos con 7 a 10 personas. Este tiempo es suficiente ya que las reuniones más largas que esta son para comenzar una iteración.
b. Quien no sea parte del equipo está fuera del círculo. Es decir, quienes son gallinas (implicados) están fuera.
c. En una pizarra se anotan los problemas y obstáculos reportados.
d. El Scrum Master borra los obstáculos si estos fueron eliminados.
e. Se debe empezar la reunión puntualmente.
f. No se permite otra discusión aparte de las tres (o cinco) preguntas. El Scrum Master tiene la autoridad de reenfocar la discusión.

g. Si otro problema amerita discusión, se realiza una segunda reunión inmediatamente después de la reunión de Scrum, donde por lo general participa sólo un grupo de los miembros del equipo.

2. Valor de las reuniones de Scrum

a. Ya que el equipo se organiza por sí mismo, estas reuniones de Scrum crean un mecanismo diario de aviso veloz al equipo acerca del estado del proyecto y las personas. De tal manera que las personas pueden tomar medidas al respecto.
b. Al momento de manifestar que es lo que se llevará a cabo para el día siguiente, se realiza un tipo de promesa social al equipo. Esto aumenta el sentido de responsabilidad.
c. Scrum se basa en el punto de vista de que el desarrollo del software da como resultado un nuevo producto creativo e impredecible, de tal modo que son necesarios métodos empíricos, es decir, métodos que uno aprende a utilizar y a combinar a través de la experiencia, más que definidos, las reuniones de Scrum brindan un mecanismo de medición frecuente, necesario para los métodos empíricos.
d. Estas reuniones dan paso a un debate diario para actualizar las tareas y eliminar los obstáculos.
e. Con la pregunta “e” del apartado de reuniones de Scrum se consigue que la gente mejore continuamente.
f. Los conceptos de lenguaje compartido, valores y prácticas ayudan a un equipo de desarrollo. Estos se crean y refuerzan en el Scrum diario.

Además de estos principios existen otros [13,16]:

a. Equipos autogestionados.
b. Una vez dimensionadas las tareas no es posible agregarles trabajo extra.
c. Iteraciones de desarrollo de frecuencia inferior a un mes, al final de las cuales se presenta el resultado a los externos del equipo de desarrollo, y se realiza una planificación de la siguiente iteración, guiada por el cliente.

2.6 ROLES

Scrum identifica a todas las personas que intervienen o tienen interés en el desarrollo del proyecto en: propietario del producto, equipo, gestor de Scrum (llamado Scrum Manager o Scrum Master) y “otros interesados”. Los tres primeros grupos (propietario, equipo y gestor) son

los responsables del proyecto, los que según la comparación siguiente (y sin connotaciones peyorativas) serían los “cerdos”; mientras que el resto de interesados serían las gallinas [12].

2.6.1 Cerdos y gallinas.

Esta metáfora ilustra de forma muy gráfica la diferencia de implicación en el proyecto entre ambos grupos [12]:

> Una gallina y un cerdo paseaban por la carretera.
> La gallina dijo al cerdo: “Quieres abrir un restaurante conmigo”.
> El cerdo consideró la propuesta y respondió: “Sí, me gustaría. ¿Y cómo lo llamaríamos?”.
> La gallina respondió: “Huevos con Jamón”.
> El cerdo se detuvo, hizo una pausa y contestó: “Pensándolo mejor, creo que no voy a abrir un restaurante contigo. Yo estaría realmente comprometido, mientras que tu estarías sólo implicada”.

En la figura 2.8, se observa una representación de la metáfora de cerdos y gallinas.

Figura 2.8: Metáfora del Cerdo y la Gallina
Simoni, 2.8

En la tabla 2.1, se realiza la identificación de los interesados en el desarrollo del proyecto [12].

Tabla 2.1: Identificación de los interesados en el desarrollo del proyecto

COMPROMETIDOS (cerdos)	IMPLICADOS (gallinas)
- Propietario del producto	- Otros interesados
- Equipo de desarrollo	- (Dirección general
- Scrum Master	- Dirección comercial
	- Marketing Usuarios, etc.

A continuación se describen brevemente los roles de la tabla 2.1 [9]:

1. **Propietario del producto:** Es responsable de obtener el mayor valor de producto para los clientes, usuarios y resto de implicados. Algunas de las tareas que debe realizar son las siguientes:
    a. Representar los intereses de la organización.
    b. Colaborar con el equipo de desarrollo diariamente.
    c. Comunicar los requerimientos del producto.
    d. Ordenar los requerimientos según su valor para la organización y el riesgo.
    e. Supervisar los incrementos entregados.
2. **Equipo de desarrollo:** Grupo o grupos de trabajo que desarrollan el producto, los cuales deben:
    a. Estimar el costo de desarrollo y comunicar necesidades de intercambio de funcionalidades.
    b. Determinar sus tareas de manera independiente.
    c. Mantener estándares y las buenas prácticas.
    d. Responsabilizarse de las demostraciones al final de cada iteración.
    e. Encargarse de construir el producto en incrementos.
    f. Comprometerse en grupo a que el equipo termine el “backlog” para el Sprint.
3. **Scrum Master:** Gestor de los equipos, es responsable de la aplicación y correcto funcionamiento del método Scrum y de la productividad del equipo de desarrollo [12]. Además debe:
    a. Conocer las prácticas y filosofía de Scrum.
    b. Asegurarse de que el equipo siga el proceso.
    c. Hacer de facilitador entre el responsable del producto y el equipo de desarrollo.
    d. Proteger al equipo de desarrollo de impedimentos que afecten su productividad.
    e. Equivaler al gerente de proyecto.
4. **Otros interesados o roles externos:** Poseen las siguientes características:
    a. Generalmente son los directivos de la organización.
    b. Mínima participación directa en el proyecto.
    c. Controlan asuntos de procesos a través del Scrum Master.
    d. Controlan asuntos del producto a través del responsable del producto.

En la figura 2.9, se observan los diferentes roles que existen en Scrum.

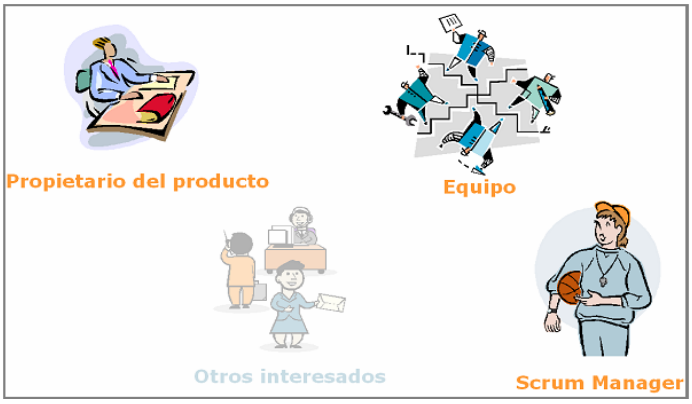

Figura 2.9: Roles en Scrum
Palacio, 2.9

Además dentro de este método se definen otros tres roles, de los cuales los primeros dos son implícitos y el tercero de apoyo [18]:

1. Cliente: Participa en las tareas relacionadas con los ítems del registro.
2. Usuario: La dimensión del equipo total de Scrum no debería ser superior a diez ingenieros. El número ideal es siete, más o menos dos, una cifra canónica en ciencia cognitiva. Si hay más, lo más recomendable es formar varios equipos. No existe una técnica oficial para coordinar equipos múltiples, pero se han documentado experiencias de hasta 800 miembros, divididos en Scrums de Scrum, definiendo un equipo central que se encarga de la coordinación, las pruebas cruzadas y la rotación de los miembros.
3. Administrador. Está a cargo de las decisiones fundamentales y participa en la definición de los objetivos y requerimientos. Por ejemplo, selecciona al Dueño del Producto, evalúa el progreso y reduce el registro de acumulación junto con el Scrum Master.

2.7 PRÁCTICAS CLAVES EN SCRUM

Scrum realiza el control de la evolución del proyecto de manera empírica y adaptable, empleando las siguientes prácticas de la gestión ágil:

1. **Revisión de las Iteraciones:** Al finalizar cada iteración (normalmente 30 días) se lleva a cabo una revisión con todas las personas implicadas en el proyecto [9,12,14].
2. **Desarrollo incremental:** Durante el proyecto, las personas implicadas no trabajan con diseños o abstracciones. El desarrollo incremental implica que al final de cada iteración

se dispone de una parte del producto funcionando de tal manera que se puede inspeccionar y evaluar [9,12,14].

3. **Desarrollo evolutivo:** Los modelos de gestión ágil se emplean para trabajar en proyectos donde existe incertidumbre e inestabilidad en cuanto a los requisitos. Predecir cómo será el producto final en las fases iniciales del proyecto, y además desarrollar el diseño y la arquitectura del producto sobre dicha predicción no puede considerarse realista, pues lo que sucede en la práctica es que las circunstancias obligan a remodelarlo muchas veces. Para qué predecir los estados finales del diseño o arquitectura si van a estar cambiando. En Scrum se toma como punto de partida a la inestabilidad, y se adoptan técnicas de trabajo para permitir la evolución hacia los estados finales manteniendo la calidad de la arquitectura que se irá generando durante el desarrollo. El desarrollo Scrum va generando tanto el diseño como la arquitectura final de forma evolutiva durante todo el proyecto [9,12,14].
4. **Auto-organización:** Durante el desarrollo de un proyecto surgen muchos factores impredecibles en todas las áreas y niveles. La gestión predictiva asigna la responsabilidad de resolución al gestor de proyectos. En Scrum los equipos son auto-organizados y no así auto-dirigidos, tiene una cierta capacidad de decisión en situaciones que consideren oportunas.
5. **Colaboración:** Las prácticas y el entorno de trabajo ágiles facilitan la colaboración del equipo. Ésta es necesaria para que funcione la autoorganización como un control eficaz donde cada miembro del equipo colabora de forma abierta con los demás, según sus capacidades y no según su rol o su puesto [9,14].
6. **No se adiciona trabajo externo a la iteración una vez elegida:** Se planifica todo el trabajo a realizar antes de comenzar la iteración, y en el caso en que se necesite más, eso se agrega en la siguiente iteración [13].
7. **Reuniones diarias de pie con preguntas especiales:** Estas reuniones son muy útiles, ayudan a mantener una buena relación entre el cliente y el equipo de desarrollo. Se deben realizar todos los días, durante 15 minutos cada día [16].
8. **Evita procesos:** No define procedimientos detallados, se deben evitar procesos que no lleguen a ser cumplidos, es decir, que se eviten planificar procesos que no serán posibles de implementar [16].
9. **Equipos de siete:** Es el número de integrantes más recomendable aún en proyectos grandes ya que en estos se forman equipos múltiples [16].

10. **Ambiente común:** Es ideal que el equipo trabaje junto en un ambiente de desarrollo común, lo cual es preferible a oficinas o cubículos. Los ambientes separados también son útiles en distintas ocasiones. De cualquier forma si el equipo está separado geográficamente, la comunicación se realiza por teléfono [16].

2.8. VALORES DE SCRUM

Scrum es una “carrocería” para dar forma a los principios ágiles, constituye una ayuda para organizar a las personas y el flujo de trabajo; como lo pueden ser otras propuestas de formas de trabajo ágil, entre ellas**:** Crystal Methods, DSDM, etc. La carrocería sin motor no funciona, es decir, que Scrum sin los valores que dan sentido al desarrollo ágil no podrá ir adelante, a continuación se listan estos valores [16].

1. **Responsabilidad y compromiso:**
   El equipo de desarrollo se hace responsable de un objetivo definido para una iteración, este equipo tiene la autoridad y autonomía para decidir por ellos mismos la mejor manera de llevarlo a cabo.

   El equipo y el Scrum Master se responsabilizan de evitar la introducción de tareas nuevas durante una iteración, evitan dirigir el equipo y trabajan para proveer los recursos y eliminar rápidamente aquello que el equipo reporte innecesario.

   El propietario del producto es responsable de definir y priorizar el backlog del producto, guiar la elección de los objetivos de la siguiente iteración y revisar y realizar la retroalimentación de el resultado de cada iteración.

2. **Enfoque:**
   El equipo de desarrollo debe ser capaz de enfocarse en los objetivos establecidos en la iteración, sin distracción. Así el Scrum Master se enfoca en proveer recursos, eliminar aquello que obstruye el desarrollo efectivo, y evita las interrupciones al equipo con tareas adicionales.

3. **Apertura:**

   El producto backlog abiertamente accesible hace visible el trabajo y las prioridades. Los Scrum diarios hacen visible el estado y las responsabilidades individuales y generales. La velocidad se hace visible con el gráfico backlog.

4. **Respeto:**

   Los miembros de manera individual en un equipo son respetados por sus fortalezas y debilidades y no discriminados por fallas en la iteración. El equipo entero más que el Scrum Master adopta la actitud de resolver problemas individuales eligiendo de un grupo de soluciones.

5. **Coraje:**

   El Scrum Master tiene el coraje de planificar y guiar la adaptatividad y confiar en los individuos así como en el equipo evitando decirles como llevar a cabo la iteración. El equipo tiene el coraje de tomar la responsabilidad de auto dirigirse y auto manejarse.

2.9 CICLO DE VIDA DE UN PROYECTO SCRUM

El ciclo de vida Scrum está compuesto de cuatro fases: planeación, arquitectura, desarrollo y cierre.

El momento en que el proyecto alcanza una relativa madurez la visión de la fase de planeación está terminada, ya que los objetivos han sido satisfechos y se comienza con la arquitectura y se priorizan los requerimientos con los que se trabajará en las primeras iteraciones, esta fase es completada tras la realización de iteraciones en la fase de desarrollo.

Se alcanza calidad confiable en cada iteración, de manera más notable en la última de ellas donde existen menos tareas nuevas para ser desarrolladas [16].

La tabla 2.2 muestra de manera resumida de que trata cada fase del ciclo de vida Scrum [16].

Tabla 2.2: Ciclo de vida Scrum

	PRE-GAME PLANEACIÓN	**ARQUITECTURA**	**DESARROLLO**	**CIERRE**
PROPÓSITO	Establecer la visión. Plantear las expectativas y financiación segura.	Identificar más requerimientos y priorizar lo suficiente para la primera iteración.	Implementar un sistema listo para ser expuesto luego de una iteración de 30 días (Sprint)	Operacional Despliegue
ACTIVIDAD	Escribir la visión, presupuesto, el backlog inicial del producto y estimar los objetos.	Planeación. Diseño y prototipos exploratorios.	Reunión de planeación del Sprint en cada iteración. Definición del Sprint. Backlog y estimaciones. Reunines de Scrum diarias. Revisión del Sprint.	Documentación Entrenamiento Marketing y ventas

A continuación se describen las fases del ciclo de vida Scrum.

1. **Antes del Juego (Pre-Game):** Antes de llevar a cabo el desarrollo del proyecto, se especifica lo que se va a realizar en las iteraciones, además de la prioridad con la que se lo hará, esta fase consta de tres puntos destacables, que se describen a continuación [12,16].
   a. **Planeación:** Definición de un nuevo entregable basado en el backlog del producto, el cual ya se conoce, junto con una estimación de su horario y costo. Si se está desarrollando un nuevo sistema, esta fase consiste en la conceptualización y el análisis. Si se está mejorando un sistema existente, esta fase consiste en un análisis limitado.

      Durante la planeación del pregame todos los miembros del equipo pueden contribuir a la creación de una lista de características, casos de uso, mejoras, defectos, etc., que se hayan detectado en el backlog del producto. El dueño del producto o un representante del mismo da a conocer sus necesidades, es así que en esta sesión se genera el trabajo para la primera iteración y un poco más [16].

En esta fase se realizan las siguientes tareas [17]:

i. Recopilación de requerimientos para conformar el backlog. (p.e. Historias de Usuario.)
ii. Desarrollo de una lista comprensiva del backlog priorizando requerimientos evaluados por el cliente.
iii. Definición de la fecha de entrega del Sprint y de la funcionalidad de uno o más hitos o entregables. Selección de la iteración más apropiada para el desarrollo inmediato.
iv. Correspondencia de los paquetes del producto (objetos) con los items del backlog en la iteración seleccionada.
v. Definición de los equipos de proyecto para la construcción de un nuevo Sprint.
vi. Análisis de riesgos y de controles apropiados para los riesgos.
vii. Revisión y ajuste posible de los ítems del backlog y los paquetes..
viii. Validación o reselección de las herramientas y de la infraestructura de desarrollo.
ix. Calculo o estimación del costo de la iteración, incluyendo el desarrollo, material colateral, comercialización, entrenamiento, y salida del producto.
x. Verificación de la aprobación y del financiamiento de la gerencia o factibilidad del proyecto aprobado.

b. **Arquitectura:** Diseñar cómo los requerimientos del backlog del producto serán puestos en ejecución. Esta fase incluye la modificación de la arquitectura del sistema y el diseño de alto nivel.

En esta fase se realizan las siguientes tareas [17]:

i. Revisión de los ítems del backlog asignados.
ii. Identificación de cambios necesarios para poner los ítems del backlog en ejecución.
iii. Análisis del dominio hasta lo requerido para construir, realzar, o poner al día los modelos del dominio y para reflejar el nuevo contexto y requisitos del sistema.
iv. Refinación de la arquitectura del sistema para apoyar el nuevo contexto y requisitos.
v. Identificación de los posibles problemas o edición del sprint dentro del desarrollo o implementación de los cambios.

vi. Reuniones de revisión de diseño, cada equipo presenta acercamientos y cambios para poner los ítems del backlog en ejecución. Reasignar los cambios como sea necesario.

c. **Planeación del Sprint:** Antes de comenzar cada iteración o Sprint, se llevan a cabo dos reuniones consecutivas, en la primera se refina y se prioriza nuevamente el backlog del producto, a demás de elegir las metas de la siguiente iteración. En la segunda reunión, tanto el equipo de desarrollo como el equipo de diseño del producto se reúnen para considerar como alcanzar los requerimientos y crear un backlog del Sprint de tareas en un rango de 4 a 16 horas [16].

Esta fase proporciona la documentación del Backlog del producto, que básicamente es un formulario, y describe a través de un listado de los requerimientos del sistema, el cual se mantiene durante todo el ciclo de vida (hasta la retirada del sistema) posee la característica de ser un documento dinámico que incorpora constantemente las necesidades del sistema [4]. Claramente se observa en la figura 2.3 que este documento consta de 4 campos principales:

a. La prioridad, en la que se describe la importancia de un requerimiento frente a otro.
b. La descripción, en la que se describe de manera resumida el requerimiento.
c. La estimación, que es una cuantificación del tamaño del requerimiento.
d. Estimado por, campo en el que se menciona a la persona que hizo la estimación.

Además se puede adicionar un campo opcional indicando a que modulo pertenece cada requerimiento.

2. Juego (Game): Una vez realizada la especificación correspondiente, se lleva a cabo la elaboración del proyecto con un continuo seguimiento a cargo del mismo grupo de desarrollo, se distinguen cuatro puntos que se desarrollan a continuación [12,16].

a. **Desarrollo de Sprints:** El trabajo generalmente se organiza en 30 días de iteraciones, cada una de ellas es llamada Sprint. El Sprint es el desarrollo de la nueva funcionalidad del entregable, con respecto a: variables de tiempo, requisitos, calidad, costo, y de la competición.

   La interacción con estas variables define el final de esta fase. Hay Sprints múltiples, iterativos del desarrollo, o los ciclos, que se utilizan para desarrollar el sistema.

Durante una iteración el Scrum Master no guía al equipo en cuanto a cómo alcanzar las metas, resolver los problemas, ni planear un orden de trabajo. El equipo se refuerza teniendo la autoridad y recursos para encontrar su propia manera de resolver los problemas. Los 30 días de iteraciones, no toman en cuenta la aprobación de los recursos y la eliminación de los obstáculos. Es quizá el reto más personal para quienes adoptan Scrum.

b. **Reuniones de Scrum:** Como ya se mencionó se llevan a cabo de manera diaria durante 15 minutos, de pie y en círculo donde se formulan 3 o 5 preguntas directas.

c. **Evitar adiciones a la iteración:** Mientras se lleva a cabo una iteración, el Scrum Master no puede adicionar tareas al equipo ni al individuo, es por ello que se llevan acabo reuniones de definición de prioridades, requerimientos y metas antes de comenzar la iteración.

d. **Decisiones en una hora:** Los obstáculos que se presenten y requieren la decisión del Scrum Master se reportan en las reuniones de Scrum de tal modo que se tome una decisión inmediata o dentro de una hora.

Esta fase provee la siguiente documentación:

a. **Backlog del Sprint.** En el formulario de control del Sprint, se hace referencia al responsable, a la descripción de su tarea, el esfuerzo restante en días y el esfuerzo total realizado, el formato de este formulario es el que se observa en la figura 2.4.

b. **Control del Sprint.** Para un mejor control del Sprint se recurren a gráficas, ver figura 2.10 y 2.11, que reflejan tanto las horas pendientes, las tareas pendientes tanto a nivel individual como a nivel del equipo de desarrollo para cada Sprint.

c. **Control de Cambios.** Uno de los principios del manifiesto ágil es el siguiente: “Dar la bienvenida a los cambios”, estos cambios deben estar registrados ordenadamente en un formulario, en el que se debe especificar tanto la fecha del cambio, el autor de la versión anterior, la persona que modificará la versión anterior, la justificación del cambio y una descripción de la nueva versión; cabe recalcar que el cambio debe estar documentado conforme a lo acordado por el equipo Scrum.

El formulario debe ser elaborado por el equipo del proyecto, y puede ser variable de acuerdo al entendimiento que se tiene dentro del equipo y un formato sugerido es el que se observa en la figura 2.12.

Proyecto	SPRINT	INICIO	DÍAS
Nombre del proyecto	1	1-feb-06	20

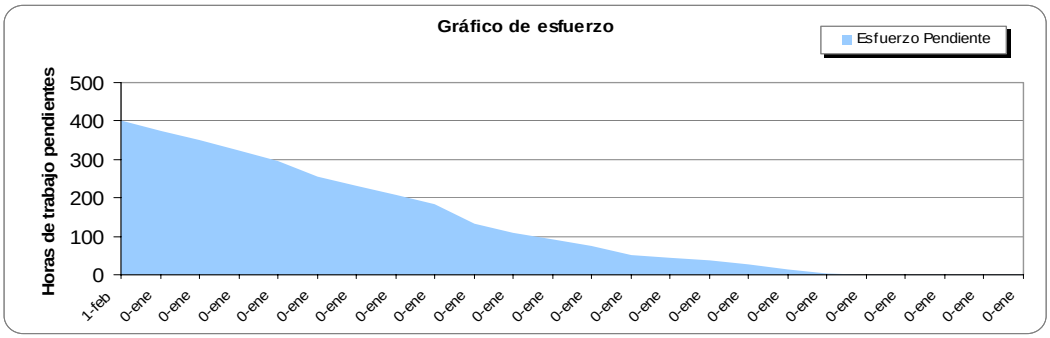

Figura 2.10: Grafica para el control del equipo Scrum
Palacio, 2.10

Figura 2.11: Grafica para el control del equipo Scrum
Palacio, 2.11

FORMULARIO DE CONTROL DE CAMBIOS					
FECHA DEL CAMBIO	APROBACION (Scrum Manager)	AUTOR DE LA VERSIÓN ANTERIOR	PERSONA MODIFICANTE	JUSTIFICACIÓN	DESCRIPCIÓN DE LA NUEVA VERSIÓN

Figura 2.12: Formulario para el control de cambios.

3. **Después del Juego (Post-Game):** Culminada una iteración, sólo resta una revisión final para así determinar los posibles cambios en la siguiente iteración, a esto se llama cierre.

   a. **Cierre:** Se realiza la preparación para la nueva iteración, incluyendo la documentación final, pre-iteración y la prueba efectuada. Al finalizar cada iteración, se lleva a cabo una reunión de revisión, máximo de cuatro horas, a la cual asiste el equipo de desarrollo, el Scrum Master, el dueño del producto y otros implicados en el desarrollo. Se presenta un demo del producto, las metas incluyen la información de las funciones, diseño, ventajas, inconvenientes, esfuerzos del equipo, y futuros problemas. Se toman en cuenta lluvia de ideas y retroalimentación del futuro, pero no se comenta al respecto, esto se hará en la siguiente reunión de Sprint de planeación. Se prohíben las presentaciones en Power Point ya que se hace énfasis en la presentación del producto [16].

En la figura 2.13, se observa de manera resumida los puntos que forman parte del método Scrum.

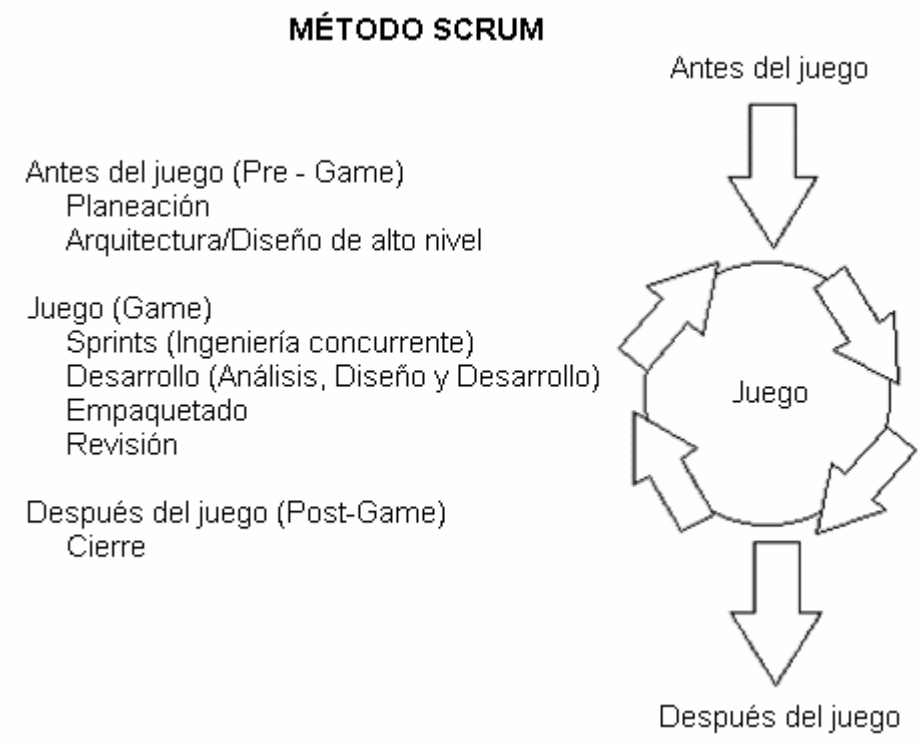

Figura 2.13: Ciclo de Vida SCRUM
Bach, 2.13

2.9 Métricas de calidad

La métrica de calidad denominada “Eficacia” de la eliminación de defectos, es adecuada para utilizarse con procesos iterativos, como SCRUM el cual es un método iterativo incremental, esta métrica es utilizada para medir la calidad de un proyecto SCRUM [19].

Eficacia de la Eliminación de Defectos (EDD)

Una métrica de la calidad que proporciona beneficios tanto a nivel del proyecto como del proceso, es la eficacia de la eliminación de defectos (EED) En particular el EED es una medida de la habilidad de filtrar las actividades de la garantía de calidad y de control al aplicarse a todas las actividades del marco de trabajo del proceso [19].

Cuando se toma en consideración globalmente para un proyecto, EED se define de la forma siguiente [19]:

$$EED = E / (E + D)$$

Donde: E= es el número de errores encontrados antes de la entrega del software al usuario final; D= es el número de defectos encontrados después de la entrega.

El valor ideal de EED es 1, y representa que no se han encontrado defectos en el software. De forma realista, *D* será mayor que cero, pero el valor de EED todavía se puede aproximar a 1 cuando *E* aumenta. En consecuencia cuando *E* aumenta es probable que el valor final de D disminuya (los errores se filtran antes de que se conviertan en defectos) Si se utiliza como una métrica que suministra un indicador de la destreza de filtrar las actividades de la garantía de la calidad y el control, el EED alienta a que el equipo del proyecto de software instituya técnicas para encontrar los errores posibles antes de su entrega [19].

Del mismo modo el EED se puede manipular dentro del proyecto, para evaluar la habilidad de un equipo en encontrar errores antes de que pasen a la siguiente actividad, estructura o tarea de ingeniería del software. Por ejemplo, la tarea de análisis de requerimientos produce un modelo de análisis qué se puede inspeccionar para encontrar y corregir errores. Esos errores que no se encuentren durante la revisión del modelo de análisis se pasan a la tarea de diseño. Cuando se utilizan en este contexto, el EED se define como [19]:

$$EED = Ei / (Ei + Ei+1)$$

Donde E*i* = es el número de errores encontrados durante la actividad *iesima* de iteración i, Ei + 1 = es el número de errores encontrado durante la actividad de iteración (i + 1) que se puede seguir para llegar a errores que no se detectaron en la actividad *i* [19].

Un objetivo de calidad de un equipo de ingeniería de software es alcanzar un EED que se aproxime a 1, esto es, los errores deberían filtrarse antes de pasarse a la iteración siguiente. Esto también puede ser utilizado dentro del proyecto para evaluar la habilidad de un equipo, esto con el objetivo de encontrar deficiencias que harán que se atrase el proyecto [19].

GLOSARIO

Ágil: Que tiene viveza y soltura, en ingeniería de software, desarrollo rápido de proyectos software

Autogestionados: Organización y dirección de algo por sí mismo

Backlog: Pila o cúmulo.

Criticidad: Característica decisiva, que debe atenderse o aprovecharse

Demo: Muestra, ejemplar.

Iteración: Repetición.

Método: Procedimiento sistemático y ordenado para realizar algo común, conjunto de reglas.

Metodología: Ciencia que estudia los métodos de adquisición de conocimientos.

Scrum: Reunión diaria de evaluación que lleva un tiempo de 15 a 20 minutos

Sprint: Iteración que dura 15 a 30 días.

Bibliografía

[1] Canós J., Letelier P., Penadés C. Metodologías Ágiles en el Desarrollo de Software. [12/11/03]. Disponible en: http://issi.dsic.upv.es/tallerma/actas.pdf. [05/05/06]

[2] Choque G. Diseño ágil y programación extrema. Universidad Mayor de San Andrés, 2005.

[3] Canós J., Letelier P., Penadés C. Desarrollo de Software. [05/05/06]; 1-4

[4] Enrich M. Crystal Metodologies. [04/02/03]. Disponible en: http://www.dsic.upv.es/asignaturas/facultad/lsi/trabajos/282002.ppt#276,3,Contenido. [18/04/07]

[5] Palacio J. Gestión de proyectos ágil: conceptos Básicos. [sf]. Disponible en: http://www.navegapolis.net/files/s/NST-003_01.pdf. [01/04/07]

[6] Sierra A. Introducción a la Programación Extrema. 2002; 3 (4): 1-9

[7] Noble J., Marshall S., Marshall S., Biddle R. Less Extreme Programming. sf. 1-10.

[8] De Simoni F. Ingeniería de Software II, Metodologías Ágiles PARTEI SCRUM. [sf]. Disponible en: http://www.2.dc.uba.ar/materias/isoft2/2005_02/clases/SCRUM_20051107.pdf [01/04/07]

[9] Giraldo J. Asisnet Ltda. 2006 . Scrum: Gerencia ágil de proyectos de tecnología. [03/06]. Disponible en: http://www.acis.org.co/fileadmin/Base_de_Conocimiento/IV_JornadaGerencia/JorgeGiraldo_IVJGP.pdf. [07/03/07].

[10] Palacio J. El nuevo escenario. [sf]. Disponible en: http://www.navegapolis.net/files/s/NST-002_01.pdf. [04/04/07].

[11] Palacio J. Visión general de Scrum. [sf] Disponible en: http://www.navegapolis.net/files/presentaciones/scrum_general_10.pdf. [07/03/07].

[12] Palacio J. “El modelo Scrum”. [sf]. Disponible en: http://www.navegapolis.net/files/s/NST-010_01.pdf. [07/03/07].

[13] Kroll P., MacIsaac B. Agility and Discipline Made Easy: Practices from OpenUP and RUP, USA: Addison Wesley Professional (2006).

[14] Gómez M., Gabardina J. Introducción Scrum. [24/11/06]. Disponible en: http://www.fi.uba.ar/materias/7546/material/Introduccion%20a%20SCRUM%20v1.pdf. [07/03/07].

[15] Leone C., Passerini N., Brey G., Rosso L. Metodologías Iterativas de Desarrollo. [sf]. Disponible en http://apit.wikidot.com/local--files/start/01_apit_metodologias.pdf. [17/03/07].

[16] Larman G. Agile & Iterative Development, USA: Addison Wesley. 2003

[17] Bach J. SCRUM. [10/95]. Disponible en: http://www.controlchaos.com/old-site/scrumwp.htm. [12/04/07].

[18] Reynoso C. Métodos Heterodoxos en Desarrollo de Software. [04/04]. Disponible en: http://www.microsoft.com/spanish/msdn/arquitectura/roadmap_arq/heterodox.asp [17/03/07].

[19] González H. Las Métricas de Software y su Uso en la Región. [07/05/01]. Disponible en: http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/gonzalez_d_h/. [08/05/07].