

# **Scrum** en Ingeniería de Software

Dario Palminio

2015

Palminio, Dario Andrés

Scrum en ingeniería de software - Dario Andrés Palminio.

- 1a ed. - Córdoba : el autor, 2015.

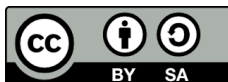
ISBN 978-987-33-8286-4

1. Ingeniería de Software. I. Título.

CDD 005.1

Versión 1.0.0-SNAPSHOT (24/07/2015)

Los artículos y las ilustraciones de este libro se distribuyen bajo  
una licencia Creative Commons by-sa 4.0



[http://creativecommons.org/licenses/by-sa/4.0/deed.es\\_AR](http://creativecommons.org/licenses/by-sa/4.0/deed.es_AR)

Pueden ser copiados, distribuidos y modificados bajo las  
condiciones de reconocer a los autores y mantener esta licencia  
para las obras derivadas.

Gracias a todos los hombres y mujeres que van a trabajar cada día y llevan con ellos su calor humano; contribuyendo positivamente a su equipo, a su organización y al mundo por ser lo que son, haciendo lo que hacen, cuando lo hacen.



## Prefacio

Quien escribe se propuso ofrecer una guía para el conocimiento e implementación de una manera de trabajar y de gestionar proyectos de Ingeniería de Software llamada Scrum. Se propone explicarle al lector el sistema Scrum de un modo integrador desde diferentes perspectivas y fuentes de conocimiento Scrum. De este modo se intenta proporcionar un marco integral que incluya los principios filosóficos, estructura y procesos de Scrum y aspectos complementarios. Se busca ayudar a los equipos a avanzar de intentar emplear Scrum a ejecutar Scrum correctamente para lograr alcanzar los resultados que aún no se han alcanzado. También se pretende brindar un material de apoyo para: facilitadores que emprendan el coaching de equipos, equipos de desarrollo de software e interesados en la Ingeniería de Software.



# Índice

<b>1</b>	<b>Introducción</b>	<b>11</b>
1.1	Consideraciones iniciales . . . . .	11
1.1.1	Definición . . . . .	11
1.1.2	Sobre si es una Metodología . . . . .	11
1.1.3	Ámbito de aplicación . . . . .	12
1.1.4	Visión general . . . . .	13
<b>2</b>	<b>Origen</b>	<b>15</b>
2.1	Origen histórico . . . . .	15
2.2	Origen causal . . . . .	17
2.2.1	Problema . . . . .	17
2.2.2	Filosofía criticada . . . . .	18
2.2.3	Metodología criticada . . . . .	19
<b>3</b>	<b>Filosofía Scrum</b>	<b>27</b>
3.1	Mentalidad y modelos mentales . . . . .	28
3.2	Principios . . . . .	29
3.3	Manifiesto Ágil . . . . .	30
3.3.1	Valores del Manifiesto Ágil . . . . .	31
3.3.2	Principios del Manifiesto Ágil . . . . .	32
3.4	Valores de Scrum . . . . .	34
3.5	Principios de Scrum . . . . .	35
3.6	Ideas filosóficas relacionadas . . . . .	38
3.6.1	Emergentismo . . . . .	38
3.6.2	Sinergia . . . . .	39
3.6.3	Ley de Linus . . . . .	40

3.6.4	Sabiduría de multitud . . . . .	40
3.6.5	Arquitectura emergente . . . . .	41
3.6.6	Prácticas emergentes . . . . .	42
<b>4</b>	<b>Núcleo del Sistema Scrum</b>	<b>45</b>
4.1	Estructura del Sistema Scrum . . . . .	45
4.2	Sistema de Roles . . . . .	47
4.2.1	Scrum Master . . . . .	47
4.2.2	Product Owner . . . . .	47
4.2.3	Equipo de Desarrollo . . . . .	48
4.3	Proceso Scrum . . . . .	48
4.4	Flujo de artefactos . . . . .	49
4.5	Reglas y Consideraciones . . . . .	50
<b>5</b>	<b>Gestión de Proyectos</b>	<b>53</b>
5.0.1	Proyecto Scrum . . . . .	53
5.0.2	Triángulo de la Gestión de proyectos . . . .	54
5.0.3	Planificación de entregables . . . . .	54
5.0.4	Medición y métricas . . . . .	55
<b>6</b>	<b>Escalamiento</b>	<b>57</b>
6.1	Equipos de características . . . . .	57
6.2	Equipos de componentes . . . . .	59
6.3	Equipos mixtos . . . . .	60
6.4	Scrum de Scrum . . . . .	60
<b>7</b>	<b>Complementos de Scrum</b>	<b>61</b>
7.1	Técnicas de Comunicación . . . . .	62
7.1.1	Gestión visual . . . . .	62
7.1.2	Tableros Scrum/Kanban . . . . .	62
7.2	Malas prácticas . . . . .	64
<b>8</b>	<b>Glosario y Acrónimos</b>	<b>69</b>



# Lista de figuras

1.1	Modelo de dominios de Marco Cynefin . . . . .	13
1.2	Mapa mental sobre Scrum . . . . .	14
2.1	Modelo Cascada de desarrollo . . . . .	20
2.2	Metodología de Gestión de Proyectos Clásica PMI	22
2.3	Ciclo de vida en una Gestión de Proyectos tradicional	25
2.4	Metodología Cascada criticada por Scrum . . . . .	26
3.1	Modelo de emergentismo . . . . .	39
3.2	Modelo del espectro del tipo desarrollo de software	42
4.1	Diagrama del Núcleo del Sistema Scrum . . . . .	46
4.2	Diagrama del Sistema de Roles Scrum . . . . .	51
4.3	Diagrama de Flujo de Datos del Proceso Scrum . .	52
4.4	Diagrama de Flujo de Stock de artefactos Scrum .	52
5.1	Proyecto Scrum . . . . .	54
5.2	Triángulo de Gestión de Proyectos Scrum . . . . .	55
6.1	Equipos de Características y Equipos de Componentes. . . . .	58
6.2	Ejemplo de equipos mixtos. . . . .	60
7.1	Tablero Kanban para Scrum . . . . .	63
7.2	Tablero Scrum . . . . .	64



# Capítulo 1

## Introducción

### 1.1 Consideraciones iniciales

#### 1.1.1 Definición

Scrum es un marco de trabajo (framework) para construir y mantener productos complejos [SBOK, 2013] [Scrum Alliance, 2015]. Scrum funciona como una implementación del ciclo de mejora continua de Deming (PDCA) y como una implementación de los principios ágiles y principios Scrum. Hay que tener en cuenta que Scrum no es exactamente un proceso íntegro, metodología completa o una técnica para construir productos; sino que, es un marco de trabajo dentro del cual se pueden emplear varias técnicas y procesos [Agile Atlas, 2012].

#### 1.1.2 Sobre si es una Metodología

Hay quienes consideran que Scrum no es una metodología, entre otras cosas porque no especifica exactamente el cómo se hacen las cosas, sino que dice el qué hacer. Sin embargo, hay autores y guías que tratan a Scrum como metodología [SBOK, 2013]. De hecho en el informe original de Ken Schwaber se habla de metodología [Ken Schwaber, 1995]. En consecuencia, se puede encontrar en numerosa bibliografía que el marco de trabajo Scrum

puede ser denominado Metodología de Desarrollo Scrum o Metodología de Gestión de Proyectos Scrum. En el primer caso puede deberse a que se puede considerar una metodología como un proceso de desarrollo iterativo e incremental de productos [Ken Schwaber, 1995]. Y en el segundo caso porque se puede considerar que es una alternativa a la gestión clásica de proyectos propuesta por metodologías como la Metodología de Gestión de Proyectos PMI. En este último sentido podemos recordar la definición original de Ken Schwaber: "Scrum es una metodología de gestión, mejora y mantenimiento de un sistema existente o prototipo de producción" [Ken Schwaber, 1995].

Por otro lado, considerando que Scrum define roles, artefactos, actividades, flujo del ciclo de actividades Scrum [Agile Atlas, 2012], reglas y algunas sugerencias de implementación como, además, al definir el flujo del ciclo de Scrum o flujo de trabajo define parcialmente un cómo, en el cual incluye una secuencia básica de cosas a hacer; por eso, y sin ser puristas, se puede considerar como una forma de metodología de trabajo y de gestión. O sea que puede funcionar como una metodología a alto nivel o plataforma de trabajo sobre la cual pueden funcionar otras metodologías, más específicas de producción y desarrollo, y otras técnicas y procesos. Por este motivo, Scrum puede ser adaptado a diversas empresas y organizaciones que trabajen con metodologías diferentes pero compatibles con los lineamientos de Scrum (sus valores y principios) y del Movimiento Ágil (filosofía ágil). Se puede usar Scrum y a su vez utilizar técnicas de otras metodologías para implementar sus actividades y sugerencias. O sea que cuando se usa Scrum se hace una aproximación empleando diversas técnicas y, posiblemente, otras metodologías.

### 1.1.3 Ámbito de aplicación

Relacionado a su ámbito de aplicación se puede decir que Scrum no es un marco de trabajo orientado a implementarse en cualquier dominio y contexto. Scrum está pensado para proyectos bajo "dominios complejos" [Snowden 2007] donde existe un grado alto de incertidumbre y baja predictibilidad (ver figura 1.1). O sea que es útil en ámbitos con requisitos inciertos y riesgos técnicos altos. Nos permite encontrar prácticas emergentes en dominios comple-

jos, como por ejemplo en la gestión de proyectos de innovación [Martin Alaimo, 2014]. Está orientado a contextos que necesitan niveles altos de creatividad, innovación, interacción y comunicación. Por este motivo, es bastante empleado en la industria de software, ya que en la misma existen contextos específicos de alta complejidad e incertidumbre con necesidad de creatividad e innovación. Pero también se utiliza en otras industrias con dominios de problemas de complejidad semejante. Por ejemplo ha sido empleado en: educación, organizaciones de campañas publicitarias, industria de productos de innovación, empresas de editoriales de libros, etcétera.

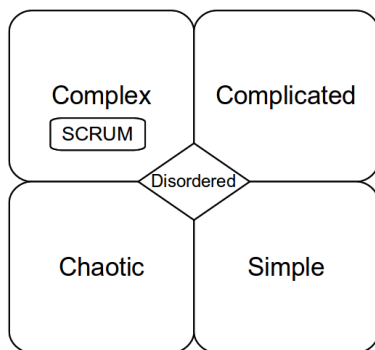


Figura 1.1: Modelo de dominios de Marco Cynefin

#### 1.1.4 Visión general

En esta metodología se definen los principios y valores a seguir, los roles, relaciones y responsabilidades, los artefactos o entidades manejadas en el proceso de trabajo, un conjunto de reuniones o actividades en un flujo de trabajo como resume la imagen de la figura 1.2.

En los siguientes capítulos se explicarán los diferentes aspectos y características de la propuesta de este marco de trabajo con lo que al final del libro el mapa mental de la figura 1.2 quedará explicado y será fácilmente entendible.

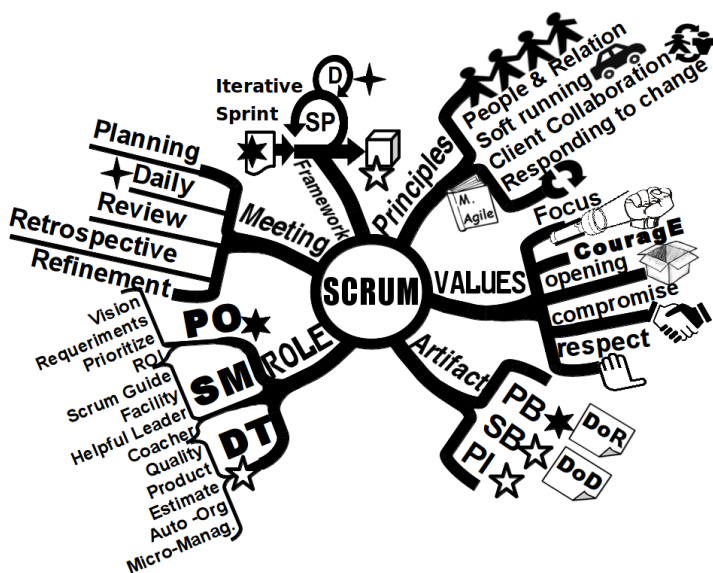


Figura 1.2: Mapa mental sobre Scrum

# Capítulo 2

## Origen

### 2.1 Origen histórico

El origen de Scrum se remonta a la década del 80. La revista gerencial "Harvard Business Review" publica un artículo de Takeuchi y Nonaka denominado: "El nuevo nuevo juego para el desarrollo de productos" [Takeuchi/Nonaka, 1986]. En el artículo se describe cómo empresas tales como Honda, Canon y Fuji-Xerox producían nuevos productos a nivel mundial utilizando un enfoque diferente al tradicional de entonces. En ese artículo se introdujo el concepto Scrum, con el término tomado del deporte Rugby, para simbolizar este nuevo enfoque basado en equipos integrales para el desarrollo de productos. Hay autores que denominan a estos conceptos originales de Scrum como "Scrum pragmático" [ScrumManager, 2014].

Una década más tarde, Jeff Sutherland y su equipo en Easel Corporation crearon el proceso de Scrum para ser utilizado en los procesos de desarrollo de software tomando los conceptos del artículo original de Takeuchi y Nonaka. Luego, en 1995, Jeff Sutherland junto con Ken Schwaber publican un informe sobre Scrum en una conferencia de Programación Orientada a Objetos OOPSLA. El informe fue formalizado con el nombre Proceso de Desarrollo SCRUM [Ken Schwaber, 1995]. Hay autores que denominan a este marco de reglas para desarrollo de software como "Scrum técnico" [ScrumManager, 2014]. Desde esa fecha,

Schwaber and Sutherland, han producido y publicado varias especificaciones para Scrum que han servido como guías y material de referencia; como por ejemplo: "Agile software development with Scrum" [Ken Schwaber, 2002] y "The Scrum Guide" [Ken/Jeff, 2013].

Luego, en los 90, Scrum paso a ser reconocidamente parte de las llamadas "Metodologías de Desarrollo de Software de peso liviano", junto a Crystal (1992), Feature Driven Development (1997), Desarrollo de Software Adaptativo (1999) y Extreme Programming (1999). Y luego del Manifiesto Ágil [Beck, 2001], promulgado en 2001 y firmado por Ken y Jeff, pasó a ser reconocida como parte del movimiento ágil y su filosofía. Ken Schwaber fue el primer presidente de la Alianza Ágil fundada tras el Manifiesto Ágil.

Desde que surgió se popularizó en en el mundo industrial, principalmente en la industria de software, miles de proyectos en todo el mundo han utilizado Scrum para el desarrollo de productos, tanto en empresas pequeñas (startups) como en multinacionales. Debido a su amplia aplicación surgieron diferentes entidades capacitadoras y certificadoras para difundir Scrum y certificar el conocimiento de quienes rinden los respectivos exámenes necesarios. La Scrum Alliance [Scrum Alliance, 2015] es un ejemplo de este tipo de organizaciones y es considerada la principal organización certificante. La Scrum Alliance fue fundada en 2002 por Ken Schwabe, Mike Cohn y Esther Derby. En 2006, Jeff Sutherland creó su propia compañía llamada Scrum.inc, sin dejar de ofrecer y enseñar a los cursos de Certified Scrum. A su vez, Ken dejó la Alianza Scrum en 2009 para fundar la Scrum.org para mejorar aún más la calidad y la eficacia de Scrum, principalmente a través de la serie Profesional Scrum ("Professional Scrum series"). Hay otras organizaciones capacitadoras y certificadoras como Scrum-Study que es una empresa (subsidiaria de VMedu) que se basa en el libro "SBOK Guide" [SBOK, 2013].



## 2.2 Origen causal

Scrum tuvo un origen causal que suele constituir el análisis crítico previo a su explicación. Scrum emerge del intento de resolver un problema y de la crítica a las consideradas causas del mismo. Scrum vino a surgir como una alternativa al modo en que se desarrollaba software y lo hizo como posible solución a los problemas que presentaba la situación en ese entonces (década del 90) de Desarrollo de Software. Por esa razón es que por lo general cuando se habla de Scrum se hace un análisis crítico previo para plantear un cambio debido a un problema. Por un lado, Scrum plantea un cambio paradigmático o filosófico que consiste en un cambio de perspectiva y de mentalidad (mindset). La otra cuestión fundamental es la metodológica en la cual se propone un cambio del proceso de desarrollo de software o proceso industrial de software. Por este motivo, para comenzar a comprender Scrum, hay que comprender cuál es el problema que viene a intentar resolver y qué es lo que se critica.

### 2.2.1 Problema

El problema principal por el que Scrum surgió como alternativa consistió en que la mayoría de los proyectos de desarrollo de software no lograban entregarse en tiempo, dentro de los costos y con las funcionalidades comprometidas. Ya en la primera conferencia organizada por la OTAN (en 1968) sobre desarrollo de software se hablaba de la "Crisis del Software" haciendo mención de los problemas recurrentes en que se veía afectado el desarrollo de software y sus resultados. En ese momento se identificaron entre diversos problemas la baja calidad del software que se desarrollaba, el no cumplimiento de las especificaciones y el código prácticamente inmantenible que dificultaba la gestión y la evolución de los proyectos. Una encuesta de cientos de proyectos de desarrollo de software empresarial indicó que cinco de seis proyectos de software se consideraban no satisfactorios [AntiPatterns, 1998], por otro lado sólo 29 de cada 100 proyectos de IT se terminaban exitosamente y el 71 por ciento de los clientes no estaban satisfechos con los resultados (Standish Groups Chaos Report 1994 - 2004). En el año 1994 el

Standish Group publicó un estudio conocido como el "CHAOS Report" [CHAOS Report, 1994] donde se mostró las siguientes tasas de fracaso en los proyectos de desarrollo de software en general:

- **Proyectos cancelados:** el 31.1 por ciento es cancelado en algún punto durante el desarrollo del mismo.
- **Proyectos insuficiente:** el 52.7 por ciento es entregado con sobrecostos, en forma tardía o con menos funcionalidades de las inicialmente acordadas.

## Causas

Los problemas detectados en los modelos tradicionales se fundamentan principalmente en lo siguiente:

- **Entorno cambiante:** Entorno altamente cambiante propio de la industria [Martin Alaimo, 2014].
- **Dependencia de Procesos rígidos:** el proceso mismo de desarrollo de software donde el resultado depende de la actividad cognitiva de las personas más que de las prácticas y controles de empleados [Martin Alaimo, 2014]. Las metodologías de desarrollo de software resultaron muy pesadas y prohibitivas para responder satisfactoriamente a los cambios de negocio.

### 2.2.2 Filosofía criticada

Como se dijo antes, con Scrum se plantea un cambio paradigmático o filosófico que consiste en un cambio de perspectiva y de mentalidad (mindset).

La filosofía basada en expertos es una de las criticadas. Normalmente solemos favorecer la opinión de los expertos, pues consideramos que sólo una persona con experiencia y conocimientos es capaz de emitir juicios verdaderos o correctos en un área o materia en particular [James Surowiecki, 2005]. Pues, es sabido

que el juicio de expertos nos puede llevar a malos resultados debido, entre otras cosas, al "mecanismo de autoridad"<sup>1</sup> que hace que respetemos o sigamos las opiniones de los expertos aunque las mismas estén equivocadas.

También se critica la filosofía basada en liderazgo. En el ámbito empresarial y en cierta época, ha prevalecido la filosofía del líder. Podíamos ver innumerable cantidad de ofertas de cursos de temáticas tales como, por ejemplo, "Masters en Liderazgo" y "Coaching en Liderazgo". Se tenía la idea de que si las organizaciones no eran fuertemente lideradas y los grupos humanos carecían de un líder entonces eran propensas al desastre. En esta perspectiva prevalecía la figura del líder coercitivo y transaccional. El líder coercitivo es el autoritario que se basa en la idea de jerarquía jefe y subordinado, y en la idea de mando y control. El líder transaccional es el que se basa en la motivación de premios y castigos, y también en la idea de negociación constante.

La idea del líder junto a la idea del control generan, entre otras cosas, la idea de la "jerarquía verticalista en organizaciones centralizadas". Idea que también se rechaza en el marco contextual de Scrum. Entre otras cosas, porque se genera una cultura donde prevalece el mecanismo de autoridad, mencionado anteriormente, y restringe la capacidad de innovación y creatividad. También puede generar sistemas culturales cerrados y rígidos ante situaciones cambiantes que requieren apertura y adaptación.

### 2.2.3 Metodología criticada

La causa de los problemas y fracasos de los proyectos de software en la situación dada, en ese entonces, de Desarrollo fueron atribuidos principalmente a la Metodología Cascada (Waterfall Methodology) [Ken Schwaber, 1995]. Pero hay que tener en cuenta de qué se habla cuando se critica a la metodología cascada. Pues la metodología cascada no es el modelo cascada y en ocasiones se han atribuido, en un sentido erróneo, las causas de los problemas al modelo cascada. Pues no se puede comparar Scrum con el mod-

---

<sup>1</sup>Mecanismo de autoridad: mecanismo por el cual uno se subordina a la opinión de un líder con autoridad o a un experto por su autoridad. El experimento de Milgram corrobora este fenómeno.

elo Cascada porque Scrum no es exactamente un modelo y ofrece soluciones a aspectos que el modelo cascada no ofrece. Aunque si, el modelo cascada, es parte de lo que se podría considerar una Metodología Cascada.

### Modelo en Cascada

El Modelo en Cascada (Waterfall Methodology) [Ken Schwaber, 1995] es un Modelo Secuencial de Procesos para la ingeniería de software presentado por Winston Royce en 1970, aunque ya se venía desarrollando desde antes. Para algunos críticos, el Modelo en Cascada, se convirtió en el modelo metodológico más utilizado dentro de la industria en un período de tiempo. Pero hay que considerar que el modelo solo abarca al sistema de producción o sistema de desarrollo (ver "Development Process" en la figura 2.4) de la industria, no al de gestión, y es simplemente un modelo, no una metodología.

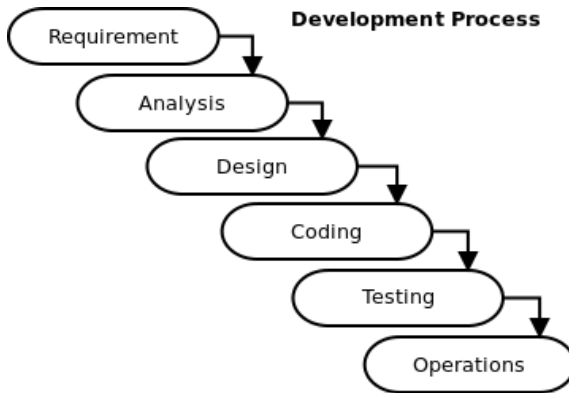


Figura 2.1: Modelo Cascada de desarrollo

En un sentido lato y ortodoxo se puede decir que el Modelo en Cascada refleja un proceso lineal y secuencial de un conjunto de procesos o fases independientes dentro de un proyecto. Las fases son: requerimientos (requerimiento del sistema y requerimientos del software), diseño, codificación, pruebas y operación. Según

esto y siempre cuando el proceso de desarrollo de un proyecto conste de solo la secuencia de estas fases sin repetición, la principales críticas como desventajas que se hacen son:

1. **Previsión:** Al tener una face de requerimientos única al comienzo, el producto final es anticipado de antemano [Scrum-Institute, 2015]. Esto requiere de cierta previsión y certidumbre inicial. La previsión es acorde a una mirada más tradicional y analítica que considera que el diseño se puede planificar en detalle al comienzo del proceso de desarrollo. A este método y práctica se lo llama BMUF que significa "Diseño Inicial Grande" o "Gran Modelado al Inicio" y con él se establece la práctica de realización explícita de Diseño Arquitectónico Completo al Inicio [Wiley/Sons, 2002]. Es el mandato técnico de crear modelos integrales de los requisitos para un sistema, el análisis de esos requisitos, una arquitectura que cumple esos requisitos y, finalmente, un diseño detallado antes de implementar el sistema. Lo que sucede es que cierta previsión y certidumbre inicial y necesaria, en proyectos complejos y cambiantes, es poco factible que suceda.
2. **Requerimientos no necesarios:** Requerimientos elicitados al comienzo del proyecto y luego implementados nunca serán completamente necesitados por el cliente [Scrum-Institute, 2015]. O sea que pueden haber requerimientos irrelevantes o innecesariamente implementados, ya sea porque el cliente dejó de tener la necesidad en el transcurso del proyecto o porque la incertidumbre inicial generó una mala elicitación de los mismos.
3. **Fases separadas:** Cada fase es estrictamente separada [Scrum-Institute, 2015]. Por ejemplo una vez que se encuentra completa la fase de requerimientos se procede a una firma de aprobación o "sign-off" que congela dichos requerimientos, y es recién aquí cuando se puede iniciar la fase de diseño, fase donde se crea un plano de modelo o "blueprint" del mismo para que, luego, los programadores lo codifiquen y se prosiga con las pruebas y finalmente el despliegue en operación. Pues en entornos altamente cambiante, propio de la industria de software, esta

forma secuencial estricta hace del proceso de desarrollo un proceso “pesadas” (estanco y burocrático) y prohibitivo para responder satisfactoriamente a los factores cambiantes de negocio [Martin Alaimo, 2014].

## Gestión de Proyectos Clásica

Bajo el marco Scrum se critica el uso de la gestión de proyectos tradicional o clásica (ver figura 2.2) en proyectos de dominios complejos y con dinámica de requerimientos cambiantes. Se atribuye parte de los fracasos a las siguientes causas:

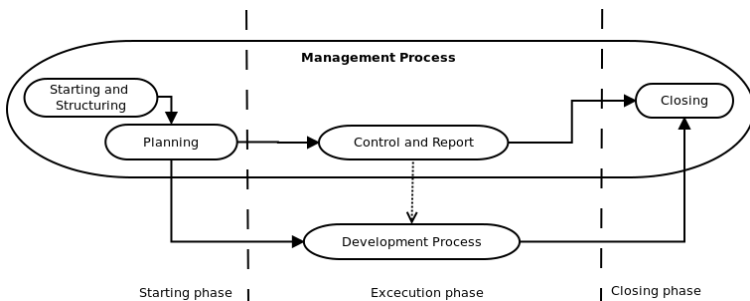


Figura 2.2: Metodología de Gestión de Proyectos Clásica PMI

1. **Planeación predictiva:** La planeación predictiva fue impulsada por el PMI quien ha adoptado el enfoque predictivo y mecanicista para la gestión de proyectos. Planeación predictiva es cuando el alcance del proyecto, el tiempo y el costo requerido para el proyecto se determina lo antes posible, en la fase de inicio del proyecto, y luego durante la ejecución del proyecto se busca seguir y respetar el plan hasta la fase de cierre (ver figura 2.3). El éxito (o rendimiento) del enfoque predictivo ha sido menos del 50 por ciento (en tiempo, en la fecha y con la funcionalidad deseada) y Scrum se presenta como alternativa para mejorar esta tasa [Ken Schwaber, 2011]. El enfoque predictivo es útil para producción de volumen alto y fabricación de bajo coste. Sus beneficios resultan de

reducir la imprevisibilidad del espacio del problema a través de la estandarización y la repetición. La planificación perfecta, la formación y la repetibilidad son las claves. Planificar y luego hacer una y otra vez. La productividad se optimiza a través de procesos de flujo de trabajo perfecto e invariable, y se optimiza el uso de los recursos (personas o máquinas). Pero esta metodología no se ajusta a proyectos donde hay más novedad que repetibilidad y donde las tecnologías, capacidades y creatividad de las personas son cambiantes. En estos casos hay alta probabilidad de que la predictividad fracase [Ken Schwaber, 2011].

2. **Planeación a largo plazo:** Debido a que el proyecto se desarrolla en forma lineal (no iterativo)<sup>2</sup> y que se hace una planificación por adelantado debido a que se considera que los cambios deben ser manejados por un sistema formal de Gestión del Cambio es que se realizan planificaciones a largo plazo. Los proyectos son planificados, en la etapa de inicio de proyecto, para efectuar entregables finales a largo plazo, en el cierre del proyecto. Los cambios que surjan en la fase de ejecución son manejados por un sistema formal de Gestión del Cambio, pero estos cambios suelen tratar de evitarse. No suele ser bien visto la introducción de cambios en la etapa de ejecución y la misma, que involucra la construcción del producto, suele abarcar un período de tiempo grande. Esto hace que se comprometan entregables a largo plazo y en forma contractual. Y el compromiso contractual efectuado al inicio del proyecto es difícil de cumplir cuando la realidad del desarrollo no se ajusta a lo planificado debido a la alta tasa de cambios resultado de la incertidumbre y a la variabilidad de las necesidades del cliente en proyectos de software o de dominio complejo. Por otro lado, si se comprueba el retorno de la inversión solo al final del proyecto y el proyecto es prolongado en el tiempo se aumenta la probabilidad de no satisfacerla y no lograr la satisfacción del cliente y estar

---

<sup>2</sup>El modelo lineal de producción y gestión fue propuesta por primera vez por Frederick Taylor en "Principios de Administración Científica", que fue la base de la línea de montaje del modelo T de Ford.

tarde en la posibilidad de corregir o ajustar para el retorno de la inversión planificado.

3. **Gestión de recursos humanos:** En la planificación tradicional las personas son gestionadas como recursos, hasta cierto punto intercambiables, individuales y especializados que siguen planes. Sin embargo la productividad, la calidad y la creatividad es mucho mayor si la gente que hace el trabajo también lo planea [Ken Schwaber, 2011]. La alta rotación de personal que no deja madurar equipos, el foco en procesos y no en ambientes de trabajo, la evaluación de desempeño individual y no grupal, la priorización de procesos de calidad y herramientas por sobre las personas y relaciones se pueden transformar en un problema.
4. **Gerente de Proyecto:** En el marco de Scrum se considera que el papel del gerente de proyecto es contraproducente en un trabajo complejo y creativo [Ken Schwaber, 2011]. La dirección de un gerente en base a un plan puede limitar la creatividad y la inteligencia del equipo en lugar de incentivarla para resolver mejor los problemas. Si se elimina la figura de gerente de proyecto y se delegan sus actividades de gestión en otros roles, junto con un enfoque de auto-organización, se puede mejorar la productividad y la creatividad.
5. **Gestión Centralizada:** La organización de la gestión tradicional suele ser centralizada. Es centralizada porque se centra en el gerente de proyectos, en una forma de liderazgo de mando y control y en jerarquías verticalistas. Las estructuras centralizadas, en organizaciones humanas, pueden formar organizaciones mecánicas y rígidas y no permitir la emergencia de la creatividad, de nuevas prácticas y de cambios ágiles adaptativos.

## Metodología en Cascada

La combinación del "Modelo en Cascada" con la "Gestión de Proyectos Clásica" conforma la metodología e idea de desarrollo de proyectos en forma de cascada (ver figura 2.4). La idea se relaciona con



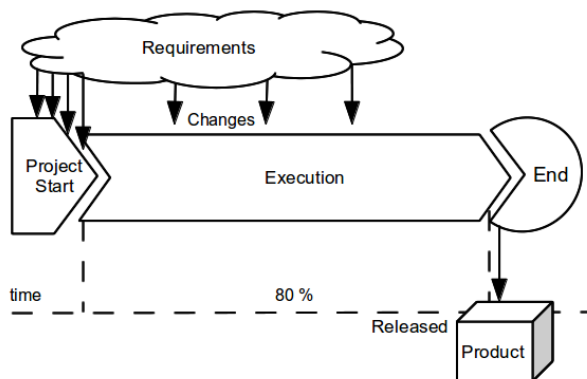


Figura 2.3: Ciclo de vida en una Gestión de Proyectos tradicional

una "carrera de relevos secuencial" en la que está incluida la administración del proyecto (proceso de gestión) y la producción del mismo (proceso de desarrollo) en una secuencia prácticamente lineal y en fases secuenciales.

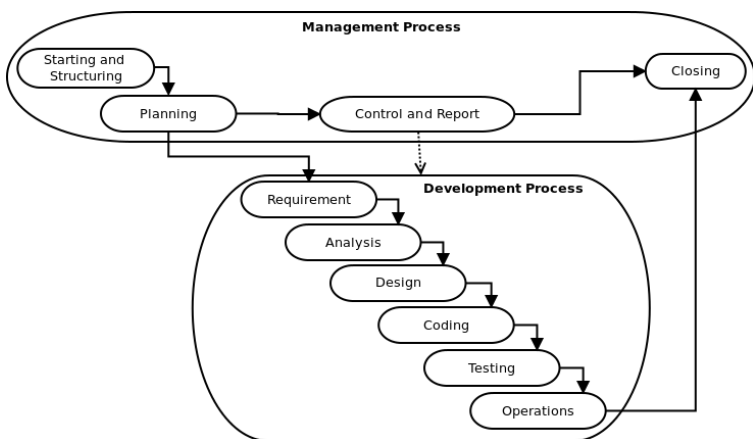


Figura 2.4: Metodología Cascada criticada por Scrum  
(Diagrama integrador del Modelo Cascada  
[Winston Royce, 1970], la Metodología Cascada  
[Ken Schwaber, 1995] y la Metodología de Gestión de Proyectos  
[PMBOK, 1996])

## Capítulo 3

# Filosofía Scrum

Discriminando todas las filosofías generales, como ideologías y concepciones religiosas, podemos decir que hay filosofías relacionadas específicamente al desarrollo de sistemas y de software. Pues, si bien ocurre que ideologías políticas influyen en los desarrolladores, líderes y arquitectos hay ideas más a fines del ámbito de desarrollo de sistemas, ideas que podemos decir que conforman filosofías influenciadoras. Estas influencias son notorias cuando vemos que se toma una decisión de usar una determinada metodología que implementa determinados principios o seguir determinados principios sin datos empíricos que sostienen su uso ni explicaciones racionales soportadas con evidencia. A veces dichos principios son como sacados de la galera o reflejan expresión de deseos. Tal como sucede cuando se aplican determinadas tecnologías o metodologías como si se tratasen de una moda o de algo aparentemente arbitrario. También esto se puede apreciar cuando escuchamos decir en una reunión de trabajo en equipo que se premiará al individuo que sobresale (filosofía individualista), que “el éxito del grupo está por encima del individual” (filosofía Ubuntu o cooperativa), escuchamos en una reunión técnica que “el software debe ser libre” (filosofía Free Software) o que solo se desarrollara software propietario, que “las mejores arquitecturas, requisitos y diseños emergen de equipos autoorganizados” [Beck, 2001] (filosofía ágil); o que “el software es un mundo de objetos” (filosofía del Paradigma

Orientado a Objetos). Las consignas o lemas organizacionales, los mantras personales o institucionales, las visiones empresariales y lineamientos institucionales también son, en muchos casos, una expresión de filosofías que condicionan a las personas y al desarrollo de software. Por ejemplo, en particular, se da con las filosofías del pensamiento sistémico, pensamiento crítico, filosofía ágil, software libre, software abierto o software propietario, que son las principales que en el mundo del desarrollo de software influyen a sus actores. Estas filosofías suelen presentar principios a seguir y estos principios guían algunas metodologías (prácticas y métodos) de desarrollo de sistemas y a gran parte de desarrolladores de sistemas. Aquí nos vamos a centrar en la filosofía Scrum y la filosofía Ágil.

### 3.1 Mentalidad y modelos mentales

Las personas que participan en el desarrollo de sistemas, diseños organizacionales o industrias de software tienen experiencias, creencias, principios, vivencias y valores que repercuten y condicionan el modo en que ellas perciben la realidad de su día a día y, en consecuencia, repercuten y condicionan su actuar, su forma de hacer, su trabajo y el resultado del mismo, sistemas hombre-máquina o software. Estas ideas, en los gerentes y desarrolladores, son modelos mentales que conforman una mentalidad, o lo que se denomina *mindset* (Masa Maeda 2012 <sup>1</sup>). O sea que la filosofía que una persona siga o adhiera está relacionada a los modelos mentales de esa persona y forja su mentalidad o *mindset* que en algún sentido lo guía.

Los modelos mentales pueden definirse como: “imágenes internas, que están profundamente arraigadas, de cómo funciona el

---

<sup>1</sup>Masa K Maeda es PhD, Founder and CEO de Valueinova USA (capacitación Scrum). Es el creador de Serious LeAP, consultor senior del Cutter Consortium en Boston, miembro del comité de dirección del Agile Testing Alliance y maestro en la Universidad de California en Berkeley. Es pionero de Lean y Kanban para trabajo de conocimiento y uno de los formadores del Lean Kanban University. Es una figura líder mundial en Agile y ha generado Serious Games de alto calibre. Previamente hizo I+D para Apple Inc en USA y para Justsystems en Japón. Obtuvo el doctorado y la maestría en Japón.

mundo, imágenes que nos limitan a las formas familiares de pensar y actuar”. Los modelos mentales abarcan cuestiones acerca de cómo vemos el mundo y de cómo actuamos en el mundo.

Tener noción de los modelos mentales es importante para entender que hay detrás de las acciones, detrás de las prácticas y de las metodologías usadas. Nos ayuda a comprender que para realizar cambios en las acciones de las personas y cambios en la forma de hacer las cosas es necesario, la mayoría de las veces, cambiar la mentalidad. Sin cambio de mentalidad puede hacerse insostenible en el tiempo un cambio de hábito, práctica o metodología. Por ese motivo, seguir principios en forma mecánica o por obediencia a la autoridad no forma convicción, y seguir una filosofía sin convicción es un primer condicionante de fracaso práctico en su implementación.

## 3.2 Principios

En la industria de sistemas y software los principios son reglas, proposiciones o normas que funcionan como máximas o preceptos que orientan la acción. O sea que un principio es como “una regla general de conducta o comportamiento” [Lawson/Martin, 2008] [SEBoK, 2014]. Un principio es como una recomendación sin el cómo se sigue la recomendación. Cómo hará para seguir la recomendación depende de usted o de quien decida seguir el principio. Por eso son la base, origen y razón fundamental sobre la cual se procede o discurre en materia de sistemas. Se suele usar en el contexto de procesos de desarrollo y metodologías como proposición que da razón, punto de partida o guía, como fundamento de un conjunto de prácticas, una metodología o paradigma de trabajo siendo las metodologías las encargadas de definir el cómo se implementan.

Cada uno sigue algunos principios en su vida como: "trabajo colaborando". También seguimos principios éticos como: “nunca mentir” o “no robar”. A semejanza de los principios éticos tradicionales, ocurre que los principios no necesariamente tienen fundamento objetivo, racional, empírico o basado en evidencias; pues, en ocasiones se comportan más como principios filosóficos que

como principios científicos. Lo cual no quiere decir que no deba buscarse que los principios en ingeniería no sean sacados de la galera o usados de forma irracional, sin justificativo razonable y sin comprobación. Es preferible aplicar principios de comprobada efectividad en su aplicación. El aplicar principios comprobados reduce la cantidad de tiempo necesaria para crear las salidas de planificación de los recursos humanos y mejora la probabilidad de que la planificación sea efectiva [PMBOK, 2004], del mismo modo aplicar principios comprobados en actividades de desarrollo y diseño de sistemas reduce tiempos de investigación para generar la salida deseada y minimiza riesgos, mejorando la probabilidad de que el desarrollo sea efectivo.

Los principios de Scrum son las pautas básicas para aplicar el marco de Scrum y guía a usarse en todos los proyectos Scrum [SBOK, 2013], proyectos en los que se aplica metodología Scrum. Los principios de Scrum se orientan a la gestión de proyectos, desarrollo de productos, trabajo en equipo y el trabajo en base a los principios ágiles. O sea que los principios Scrum tienen correlación con los principios ágiles en forma prácticamente directa [Agile Atlas, 2012]. Y los principios de Scrum están alineados a los valores de Scrum que son: Foco, Coraje, Apertura, Compromiso y Respeto.

A continuación se describen los principios y valores del Manifiesto Ágil y de Scrum.

### 3.3 Manifiesto Ágil

Los principios del desarrollo ágil se encuentran en el Manifiesto por el Desarrollo Ágil de Software o Manifiesto Ágil [Manifiesto Agile 2001], el que expone valores y principios firmado por diecisiete personas convocadas por Kent Beck [Beck, 2001]. Los principios del desarrollo ágil surgieron como principios base y originarios de métodos que estaban surgiendo como alternativa a las metodologías clásica formales (CMMI, SPICE, etc.) a las que, autores como Kent Beck, consideraban excesivamente pesadas y rígidas por su carácter normativo y fuerte dependencia de planificaciones detalladas completas y previas al desarrollo [Wiki, 2015].

### 3.3.1 Valores del Manifiesto Ágil

El Manifiesto Ágil propone los siguientes valores:

1. **Individuos e interacciones sobre procesos y herramientas:** hay que priorizar la confianza puesta en los equipos, los individuos dentro de esos equipos y la manera en que éstos interactúan en vez de seguir rígidamente procesos y herramientas. Pues, son los equipos quienes deben resolver qué hay que hacer, cómo hay que hacerlo y finalmente son ellos quienes lo hacen. Pues los equipos no deberán ser meros autómatas que reciben órdenes jerárquicas de la organización de arriba hacia abajo sino que, en lugar de eso, se espera que de abajo hacia arriba sepan resolver los problemas, ofrecer sus propios métodos de trabajo y ser hasta cierto punto autosuficientes. En este sentido, hay que delegar en ellos la identificación de qué se interpone en el camino de sus metas y la asunción de la responsabilidad de resolver todas las dificultades que se encuentren dentro de su alcance. Se les debe permitir trabajar en conjunto con otras partes de la organización para resolver asuntos que están más allá de su control.
2. **Software funcionando sobre documentación extensiva:** hay que estar orientado al producto o focalizarse en él y de este modo requerir un incremento de producto completo y funcionando como resultado final de cada ciclo de trabajo, en vez de tener que cumplir con grandes y engorrosas documentaciones y formalismos burocráticos. Ciertamente, en la construcción del producto, sea necesario realizar determinada documentación, pero es el producto concreto o funcionando lo que permite a la organización guiar al proyecto hacia el éxito. Es crucial que los equipos produzcan un incremento de producto en cada ciclo de trabajo.
3. **Colaboración con el cliente sobre negociación contractual:** en vez de tener comunicación pobre debido a restricciones contractuales, el cliente debería ser el punto de contacto principal del equipo, en colaboración de trabajo,

con los eventuales usuarios finales del producto y con las partes de la organización que necesitan el producto. Se debería ver al cliente como un miembro del equipo que trabaja colaborativamente con el resto de integrantes para decidir qué debe hacerse y qué no. Con el cliente se debería poder seleccionar el trabajo que debe realizarse a continuación, asegurando que el producto tenga el valor más alto posible en todo momento. Esto es crucial construir una fuerte colaboración con el cliente en vez de negociar contratos rígidos que obstaculizan el trabajo ágil y generan fricción con el cliente.

4. **Respuesta ante el cambio sobre seguir un plan:** el avance del trabajo o del equipo debería estar representado por un incremento de producto real y que funciona, y no por una correcta correlación y contrastación a un plan. Se debe priorizar la flexibilidad para adaptación a cambios en vez de la rigidez de seguir un plan detallado. Para ello, los equipos deberían inspeccionar lo que sucede de forma abierta y transparente buscando adaptar sus acciones a la realidad. O sea que, la planificación se debería adaptar a la realidad y al equipo y no al revés.

Se puede notar que estos valores son aplicables a cualquier tipo de organización e industria. Pues, solo el segundo valor se refiere particularmente a la industria de software y el mismo se puede reformular de forma más general como sigue: trabajar orientados al producto funcionando más que sobre una amplia y extensa documentación [Sriram Narayan, 2015].

### 3.3.2 Principios del Manifiesto Ágil

Alineados a estos valores, el Manifiesto Ágil propone los siguientes principio:

1. **Entregar valor temprano y continua:** Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor (alineado al principio de "entregar lo más rápido posible" de Lean).



2. **Apertura al cambio:** Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos Ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.
3. **Entregables frecuentes:** Entregamos software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible (alineado al principio de "entregar lo más rápido posible" de Lean).
4. **Cooperación con el cliente:** Los responsables de negocio y los desarrolladores trabajamos juntos de forma cotidiana durante todo el proyecto (alineado al principio de "potenciar al equipo" de Lean). O sea que se privilegia una cooperación constante entre miembros del equipo e interesados externos al equipo (como clientes).
5. **Personas motivadas:** Los proyectos se desarrollan en torno a individuos motivados. Hay que darles el entorno y el apoyo que necesitan, y confiarles la ejecución del trabajo (alineado al principio de "entregar lo más rápido posible" de Lean).
6. **Conversación cara a cara:** El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.
7. **Orientación al producto:** El software funcionando es la medida principal de progreso.
8. **Desarrollo sostenible:** Los procesos Ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante de forma indefinida.
9. **Excelencia técnica:** La atención continua a la excelencia técnica y al buen diseño mejora la Agilidad.
10. **Simplicidad eficiente:** La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial (alineado al principio de simplicidad). Este estilo de prácticas es parte de algo que, en metodologías ágiles, se conoce

como: "hacer todo lo posible por hacer lo menos posible" [Anacleto, 2005]. Por ejemplo en Arquitectura se puede considerar mantener una lo más simple posible. Si la simpleza se traduce en facilidad de uso de la arquitectura, facilidad para entender los conceptos involucrados y documentación necesaria, es muy probable que el nivel de productividad aumente [Anacleto, 2005]. En este sentido la simpleza de la arquitectura es un requerimiento de calidad alineado a la filosofía ágil.

11. **Equipos auto-organizados:** Las mejores arquitecturas, requisitos y diseños emergen de equipos auto-organizados (alineado al principio sistémico de emergencia).
12. **Equipos auto-reflexivos:** A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo para a continuación ajustar y perfeccionar su comportamiento en consecuencia.

### 3.4 Valores de Scrum

1. **Foco:** hay que enfocarse en sólo unas pocas cosas a la vez, trabajamos bien juntos y buscando producir un resultado excelente tratando de entregar ítems valiosos en forma pronta.
2. **Coraje:** hay que buscar sentirse apoyados y tener más recursos a disposición para promover el coraje para enfrentar desafíos más grandes. Además, para poder lograr cambios significativos en una organización que mantiene una cultura con principios y valores que entran en conflicto con los de Scrum y el Manifiesto Ágil, es necesario tener coraje para impulsar el cambio y plantarse en forma efectiva ante la resistencia al cambio. En este sentido, coraje se refiere al valor que hay que tener para no dejarse dominar ante la idiosincrasia predominante y el "status quo" que atentan contra el pensamiento Scrum.
3. **Apertura:** Hay que tener apertura para expresar cotidianamente cómo nos va, qué problemas encontramos, manifestar las preocupaciones y aceptar las sugerencias de los pares

para que éstas puedan ser tomadas en cuenta por nosotros y por los demás. La apertura requiere capacidad de aceptación y tolerancia ante la crítica y la opinión de los demás.

4. **Compromiso:** Se busca lograr compromiso para el éxito gracias a promover el mayor control nuestro sobre lo que hacemos y nuestro destino. Hay mayor probabilidad de lograr compromiso en las personas que deciden sobre lo que hacen.
5. **Respeto:** Buscamos convertirnos en merecedores de respeto a medida que trabajamos juntos, compartiendo éxitos y fracasos, llegando a respetarnos los unos a los otros y ayudándonos mutuamente.

## 3.5 Principios de Scrum

Se suele asociar a los valores del Manifiesto Ágil como principios de Scrum. Por lo que, en primera instancia, los cuatro valores ágiles son los principales principios de Scrum. Pero para no repetirlos en esta sección vamos a nombrar otros seis principios que en diferente bibliografía se suelen atribuir a Scrum. Los mismos son:

1. **Control Empírico:** El proceso empírico de control del proyecto es más efectivo que el control predictivo de largos plazos. Es más efectivo para gestionar la complejidad y obtener el mayor valor posible, basado en inspección y adaptación regular en función de los resultados que se van obteniendo y del propio contexto del proyecto. El proceso empírico permite adaptabilidad a requisitos que emergen del mismo proceso de desarrollo. Con este principio como marco, podemos usar metodologías, prácticas y técnicas, y ser nosotros (o el propio equipo de trabajo) los que a través del empirismo, determinemos la forma más adecuada de hacer las cosas para lograr los objetivos. Son los equipos de desarrollo los que deben hacer lo que sea necesario para entregar el producto esperado y aprender de su propia experiencia mediante exploración y experimentación

[UNTREF, 2014]. Es el equipo el que determina qué prácticas y herramientas les dan los mejores resultados, y así mejoran de manera continua. Los buenos equipos trabajarán constantemente en mejorar y aprender de sus experiencia. Además, se debe aprender de la experiencia de los demás leyendo libros y buscando la experiencia de personas que ya hayan venido probando algunas prácticas, o que están experimentando con nuevas potenciales mejores formas de hacer las cosas a través de la inspección y la adaptación.

2. **Auto-organización:** Los equipos auto-organizados pueden auto-gestionarse y de ellos emerge la sabiduría necesaria para la gestión de sus proyectos y actividades, y así lograr la sinergia necesaria para resolver problemas en forma ágil. Esta idea proviene de la concepción de que de un sistema social puede emerger inteligencia de grupo como puede suceder en una bandada de pájaros. Esta es una premisa aceptada en Inteligencia Artificial, pensamiento sistémico y en filosofía emergentista. Se considera que en un sistema con agentes inteligentes, a partir de reglas locales simples puede emerger inteligencia grupal colectiva o de enjambre, pues se considera que la "información local puede conducir a la sabiduría global" [Steven Johnson, 2002]. De aquí que, de la auto-organización en equipos de trabajo puede surgir inteligencia o también sabiduría según un fenómeno conocido como "sabiduría de multitud" o "wisdom of the crowd" [MIT Press, 2009] en la que la opinión colectiva de un grupo puede ser mejor que la individual de un experto. Este fenómeno, no sólo es útil a la gestión de proyectos, sino también a las actividades de estimación, pues el promedio de muchas estimaciones individuales suele estar mucho más cerca del valor real que la estimación de un experto. En lo referente al diseño se cree, como lo indica el principio 11 (once) del Manifiesto Ágil, que se logran mejores diseños y arquitecturas desde equipos auto-organizados [UNTREF, 2014]. En lo referente al liderazgo se cree que no es necesario un líder jerárquico, autoritario o experto que guíe al equipo sino que es el propio equipo el que genera su liderazgo. Según esta

perspectiva se puede prescindir de la figura de líder tradicional o jefe, se puede carecer de líder, lograr muchos líderes o tener un líder con perfil más bien de facilitador (servicial e integrador).

3. **Colaboración:** Se puede entender a la colaboración como la capacidad de reconcebir nuestras propias ideas a la luz de la de las demás [Austin, 2003] para poder lograr ideas colectivas mejores que las ideas individuales [UNTREF, 2014]. Las ideas en colaboración son resultado y mérito del equipo y no de alguno de sus integrantes. La agilidad requiere de colaboración, colaboración interna en el equipo y externa con el cliente. Colaborar con el cliente permite guiar de manera regular los resultados del proyecto de desarrollo. O sea que, la colaboración en el marco de agilidad se orienta directamente a conseguir los objetivos del cliente en un proyecto mediante el trabajo en equipo colaborativo. El trabajo en equipo con colaboración del cliente posibilita su frecuente retroalimentación y mantenerse alineado a su punto de vista y sus expectativas para satisfacer sus necesidades o los requisitos del desarrollo, por el cual el sistema producto se desarrolla con mayor agilidad. La colaboración interna requiere soltura y apertura para dejar los egos de lado e integrarse en el proceso de pensamiento colectivo, donde los problemas los resuelven todos los del equipo con sus aportes individuales. En el marco de colaboración se dejan de lados los héroes, pues los héroes no ven el gran dragón Malveau 2004 y los héroes se suelen llevar los créditos. La cooperación es la convicción de que nadie llega a la meta si no llegan todos (Virginia Burden).
4. **Priorización por valor:** Se prioriza por valor, es decir que se puede ser más efectivo si se hacen primero las tareas que suman más valor al negocio o a las necesidades del cliente. Se hace necesario prescindir de requisitos de baja prioridad antes que tener que degradar la calidad.
5. **Limitación de tiempos (time-boxing):** El trabajo limitado en periodos de tiempo ayuda en la regularidad en las

actividades. Por eso, las iteraciones de trabajo, las actividades y las reuniones deben tener un límite de tiempo y se debe buscar no sobrepasar esos límites.

6. **Desarrollo iterativo:** El desarrollo iterativo permite una construcción gradual en proyectos complejos. En cada iteración el equipo evoluciona el producto (hace una entrega incremental) a partir de los resultados completados en las iteraciones anteriores, añadiendo nuevos objetivos/requisitos o mejorando los que ya fueron completados, de manera que el cliente pueda obtener los beneficios del proyecto de forma incremental. El desarrollo iterativo permite gestionar las expectativas del cliente (requisitos desarrollados, velocidad de desarrollo, calidad) de manera regular y lograr reacción, aceptación del mercado y adaptabilidad. Permite que el cliente pueda obtener resultados importantes y útiles ya desde las primeras iteraciones. Facilita la mejora ya que al tener experiencias por períodos de iteración se puede mejorar de las experiencias de iteraciones previas y permite planificar los cambios necesarios para aumentar la productividad y calidad en iteraciones subsiguientes.

## 3.6 Ideas filosóficas relacionadas

### 3.6.1 Emergentismo

Scrum adhiere al principio de auto-organización y está alineado al Manifiesto Ágil y a su filosofía. En la filosofía ágil se toma una idea basada en el emergentismo que la podemos encontrar en el principio: "Las mejores arquitecturas, requisitos y diseños emergen de equipos auto-organizados" [Beck, 2001]. El emergentismo sostiene que el "todo puede ser más que la suma de las partes"<sup>2</sup> que significa que desde un nivel de realidad dado ( $n$ ) donde componentes se interrelacionan ( $S_1, S_2, S_3... S_n$ ) pueden emerger sistemas, propiedades o características nuevas en el nivel

---

<sup>2</sup>"The whole is more than the sum of its parts" (Ludwig Von Bertalanffy, General System theory, 1968). Esta idea del sistemismo y el emergentismo también está relacionada al holismo que sostiene la misma afirmación.

superior ( $n+1$ ) que no existen en los componentes individuales, pero que relacionados la generan (ver figura 3.1). En este sentido suele llamarse emergencia al fenómeno en el que algo emerge o es emergente, en el que una totalidad llega a existir, a una cosa nueva que posee una propiedad emergente, a un proceso nuevo que surge de algo, a una estructura que aparece de otras, a un orden que viene de otro, a una novedad cualitativa en la naturaleza que brota, a un sistema que resulta de componentes relacionados o al surgimiento espontáneo de algo nuevo.

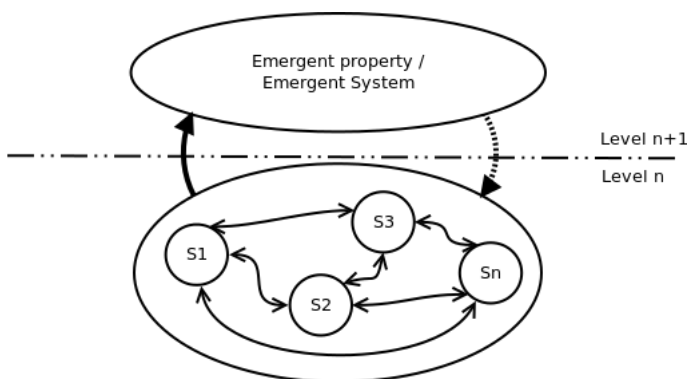


Figura 3.1: Modelo de emergentismo

En este sentido la auto-organización sucede cuando componentes, sistemas o personas se organizan sin aparente dirección o mando controlador y generan espontáneamente una forma global de orden o coordinación. Esto se da en una gran variedad de fenómenos físicos, químicos, biológicos, sociales y sistemas cognitivos. A nosotros nos incumbe principalmente lo relacionado a lo social e informático.

### 3.6.2 Sinergia

En lo social se sabe que se pueden lograr equipos de personas con una alta organización y coordinación sin la necesidad de un líder director, sin un líder que ordene y controle, e incluso sin

planificación alguna. Pues, no solo organización y coordinación se puede lograr desde equipos auto-organizados, sino que también se puede lograr sinergia. La sinergia es la prestancia extra que puede dar un equipo debido a la acción conjunta de individuos en equipo que es superior a la suma de las individualidades. Otra propiedad emergente que se puede lograr en grupos auto-organizados es la inteligencia colectiva o inteligencia enjambre que incluye a la "Ley de Linus" y a la "Sabiduría de grupo".

### 3.6.3 Ley de Linus

Se pueden formar muchos tipos de inteligencia colectiva en equipos colaborativos. Por ejemplo, lo más común en el desarrollo de software es que problemas extremadamente complejos solo pueden ser resuelto por un conjunto de personas trabajando conjuntamente en él, a pesar de que muchas veces sea una sola persona la que resuelva un determinado problema, esa persona nunca podría haberlo hecho sola. Es decir que, dada una base suficiente de desarrolladores y testers validadores colaborando, casi cualquier problema puede ser caracterizado rápidamente, y su solución ser obvia [Eric Raymond, 1997]. A esta idea de inteligencia colectiva se la denomina "Ley de Linus".

### 3.6.4 Sabiduría de multitud

La Sabiduría de grupo o de multitudes es una idea, apoyada en evidencias, de que dada "ciertas condiciones" las decisiones o predicciones tomadas colectivamente por un grupo de personas suelen ser más atinadas que las decisiones o predicciones individuales o que las que son tomadas sobre la base del conocimiento de un experto [James Surowiecki, 2005]<sup>3</sup>. Es más, a medida que el grupo es más grande, las decisiones o predicciones son mas acertadas.

Claro que para que esto suceda deben cumplirse ciertas condiciones. Para Surowiecki (escritor del libro *La Sabiduría de las Multitudes*<sup>4</sup>) las condiciones necesarias son:

---

<sup>3</sup>"The wisdom of the crowd is the collective opinion of a group of individuals rather than that of a single expert." [MIT Press, 2009]

<sup>4</sup>[James Surowiecki, 2005]



1. **Diversidad:** el grupo debe tener diversidad de perfiles para lograr una diversidad de Opiniones. Pues, si todos los del grupo piensan igual o semejante, pertenecen a la misma tribu urbana u subcultura, son de la misma profesión o tienen las mismas características de perfil profesional o psicológico, entonces es menos probable de que logren una sabiduría de grupo.
2. **Independencia:** el grupo no debería estar influenciado y las opiniones de los integrantes tampoco. Si las personas son influenciadas por el grupo se puede caer en lograr el "pensamiento de grupo" y tomar decisiones malas o irracionales. Por otro lado, si el grupo es influenciado por un actor externo al grupo la decisión grupal es dirigida.
3. **Agregación:** El sistema de decisión grupal debe ser agregativo, que significa que debe tener la capacidad de sumar o promediar las opiniones individuales. Un sistema de votación no agregativo, por ejemplo por votación de la mayoría sobre una alternativa, puede hacer prevalecer una decisión individual. Un ejemplo de agregación es cuando la decisión métrica grupal se basa en el promedio de todas la decisiones<sup>5</sup>.

### 3.6.5 Arquitectura emergente

Hay dos formas extremas de ver el desarrollo de software incluyendo al diseño de software. Una es la que sugiere que se pueden prever todas las cientos de miles de cuestiones que emergen cuando se desarrolla el software y, en consecuencia, se trata de limitar las respuestas a las mismas conduciendo un desarrollo planificado, estructurado, dirigido y liderado (el extremo izquierdo del aspecto que se muestra en la figura 3.2) [Neal Ford, 2010]. La otra forma es no anticipar nada, permitiendo que las soluciones software emerjan espontáneamente a medida que evoluciona el desarrollo en un proceso no dirigido, no liderado y descentralizado

---

<sup>5</sup>Un ejemplo de Sabiduría de multitud es cuando se le pide a muchas personas que predigan la cantidad de elementos contenidos en un frasco transparente. Es sorprendente como el promedio se acerca considerablemente al número real.

(el extremo derecho del aspecto que se muestra en la figura 3.2). En la primera opción el rol de los líderes y expertos (por ejemplo líderes de proyecto y arquitectos expertos), las reglas globales de dirección, el trabajo de comienzo y la información inicial es fundamental para el desarrollo. En el lado de la segunda opción, la orientada a fenómenos emergentes, el rol de todos los actores, el trabajo en equipo y las reglas locales de trabajo son lo principal para el desarrollo. La emergencia de arquitectura se da, entre otras cosas, cuando se crea una arquitectura inicial simple y flexible, con decisiones tomadas que no son irreversibles, y con una evoluciona en el tiempo que considera los nuevos requisitos y problemas que surjan.

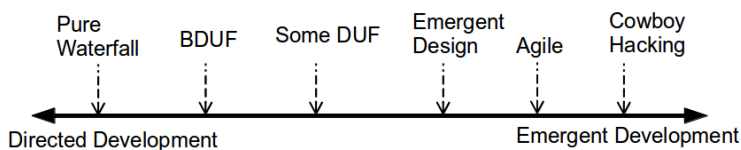


Figura 3.2: Modelo del espectro del tipo desarrollo de software (Espectro que abarca el diseño [Neal Ford, 2010])

Hay una obra literaria llamada "La catedral y el bazar"<sup>6</sup> escrita por el hacker Eric S. Raymond en 1997 que expone esta idea. La misma analiza dos modelos de producción de software: la catedral que representa el modelo de desarrollo más hermético y vertical (el lado izquierdo de la figura 3.2) y por otro lado el bazar, con su dinámica horizontal y bulliciosa (el lado derecho de la figura 3.2).

### 3.6.6 Prácticas emergentes

Scrum se promueve en un dominio de prácticas emergentes. Esto significa que no se puede mecanizar un proceso determinado bajo Scrum para obtener resultados exáctamente predecibles y repetibles en forma estandarizada. Las soluciones no son necesariamente replicables con los mismos resultados, pues sucede que ante los

---

<sup>6</sup>[Eric Raymond, 1997]

misimos problemas y requerimientos pueden surgir diferentes soluciones. Esto se debe, entre otras cosas, a que las personas no son iguales y los contextos tampoco. Para trabajar bajo Scrum es necesario seguir el núcleo de Scrum, pero implementando diferentes prácticas, técnicas y metodologías, con un grado de experimentación con fallos que intentan ser de bajo impacto. Para lograr una prestancia alta usando Scrum son necesarios niveles altos de creatividad, innovación, interacción y comunicación. En vez de determinar un proceso completo y estructurado, se intenta desarrollar un proceso flexible donde el equipo es quien va generando, sobre la marcha de un proyecto, el proceso de forma tal que el mismo emerge de un contexto relacional e iterativo de inspección y adaptación constante. Son los equipos quienes van encontrando las mejores maneras de resolver los problemas con prácticas emergentes.



## Capítulo 4

# Núcleo del Sistema Scrum

Scrum es un sistema compuesto por la filosofía Scrum (SCRUM Philosophy), como parte del sistema cultural, y tres conjuntos de componentes interrelacionados que forman el Núcleo del Sistema Scrum (ver figura 4.1). Los tres conjuntos de componentes son: roles, actividades y artefactos. Los roles conforman un sistema de roles que determina las responsabilidades de los actores integrantes, sus relaciones recíprocas, sus restricciones y la relación con los demás componentes. Las actividades conforman la parte principal del sistema de procesos Scrum (Proceso Scrum) que determina las relaciones de organización entre actividades, roles y artefactos. Y los artefactos conforman el conjunto de almacenes o componentes de trabajo. A continuación se describirán estos sistemas y sus relaciones.

### 4.1 Estructura del Sistema Scrum

Scrum está pensado como un sistema de trabajo para equipos con tres roles principales: el Product Owner, el Scrum Master y el Scrum Team.

Quando los integrantes del sistema Scrum ejercen los roles

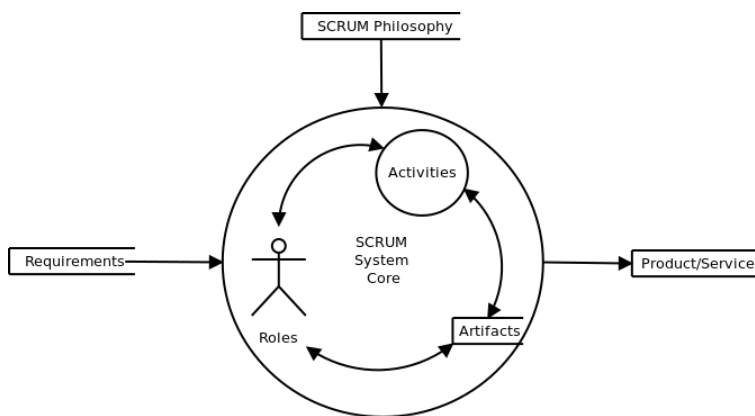


Figura 4.1: Diagrama del Núcleo del Sistema Scrum

mencionados en un flujo de trabajo o proceso Scrum, trabajan sobre tres artefactos esenciales: el Product Backlog (lo que queda por hacer), el Sprint Backlog (lo que se va a hacer) y el Incremento de Producto (lo que logramos hacer). Estos artefactos son tratados en un flujo de trabajo en el cual se construyen productos en forma incremental, en una serie de ciclos cortos de tiempo llamados Sprints.

En cada ciclo Sprint del flujo de trabajo se practican seis actividades Scrum: refinamiento de producto, planificación, reunión diaria o Daily, desarrollo, revisión de producto y retrospectiva. La actividad de refinamiento de producto (Refinement) no suele tener un nombre unificado; pues, se la suele llamar "Backlog Grooming"<sup>1</sup> (aunque no se aconseja usar la palabra grooming), Mantenimiento de Backlog o Refinamiento (Refinement). Salvo el desarrollo, las actividades se consideran reuniones o ceremonias Scrum. El desarrollo no es una reunión Scrum ya que constituye la actividad de producción del producto o servicio. O sea que es donde se produce el incremento de producto.

<sup>1</sup>No se aconseja usar el término Grooming debido a que según el diccionario Oxford English Dictionary tiene connotaciones que se presta para interpretaciones sexuales.

## 4.2 Sistema de Roles

El Sistema de Roles (ver figura 4.2) es el conjunto de roles y relaciones parte del sistema Scrum. Como se mencionó anteriormente hay tres roles principales: el Product Owner o dueño del producto, el Scrum Master o facilitador y el Scrum Team o Equipo Scrum (Miembros del Equipo de Desarrollo o Desarrolladores Scrum). También hay roles secundarios como el de Stakeholder, Vendedores y Cuerpo de Asesoramiento de Scrum. El rol Stakeholder es el más importante de los roles secundarios e incluye a los clientes, usuarios y patrocinadores.

A continuación se listan los diferentes roles principales:

### 4.2.1 Scrum Master

Para el cumplimiento del rol Scrum Master se deben tener en cuenta las siguientes afirmaciones:

- Es el responsable de que se siga Scrum.
- Necesita desempeñar su rol con coraje.
- La presencia en la Daily Scrum no es obligatoria.
- No debe estimar junto al Equipo de Desarrollo.
- La mejor manera de cumplir el rol es estar tiempo completo en este rol.
- No es gerente de proyecto, no debe gestionar al Equipo de Desarrollo y no es responsable por la planificación del proyecto.

### 4.2.2 Product Owner

Para el cumplimiento del rol Product Owner se deben tener en cuenta las siguientes afirmaciones:

- Debe colaborar con el Scrum Master y con el Equipo de Desarrollo.

- Es quien determina el mejor producto a conseguir.
- No estima ni provee estimaciones.
- Es responsable de formar una visión, comunicarla y promoverla.

### 4.2.3 Equipo de Desarrollo

Para el cumplimiento del rol Equipo de Desarrollo se deben tener en cuenta las siguientes afirmaciones:

- Es responsable de la calidad técnica del producto.
- Debe procurar su desarrollo profesional para lograr excelencia técnica.
- Debe evitar trabajar en múltiples proyectos.
- Debe priorizar y promover la comunicación cara a cara.
- Es responsable de gestionar su propio trabajo durante el Sprint.
- No debe ocultar impedimentos.

## 4.3 Proceso Scrum

En el proceso Scrum o sistema de flujo de actividades Scrum los miembros del equipo Scrum colaboran para crear una serie de Incrementos de Producto durante iteraciones de intervalos fijos de tiempo denominados Sprints. En cada iteración Sprint, se comienza por una Planificación del Sprint para producir un Backlog del Sprint a partir del Backlog de Producto, es decir un plan para el Sprint. El equipo se auto-organiza para realizar el Desarrollo, mediante reuniones Diarias de Scrum para coordinar y asegurarse de estar produciendo el mejor Incremento de Producto posible en el proceso de desarrollo del producto o servicio. Cada incremento satisface el criterio de aceptación del Product Owner y la Definición de Hecho o "Definition of Done" compartida por el



equipo para satisfacer el criterio de tarea terminada. Junto al proceso de desarrollo se hace también un Refinamiento del Backlog ("Refinement") para prepararse para la reunión de planificación del próximo Sprint. Finalizando cada ciclo se termina el Sprint con una reunión de Revisión del Sprint y luego una reunión Retrospectiva del Sprint, revisando el producto y su proceso con una perspectiva crítica y de mejora continua.

## 4.4 Flujo de artefactos

Los artefactos ítems de trabajo fluyen desde que se definen en el Backlog de Producto hasta que se transforman en incremento de producto. Los ítems de trabajo son items de valor para el cliente que comienzan su nacimiento como items de backlog o PBIs del Backlog de Producto. Debido a que el dueño del Backlog de Producto es el Product Owner, es él quien crea los PBIs, ya sea por trabajo individual o con la colaboración del Equipo de Desarrollo. Los PBIs son requerimientos que puede escribirse de diferente manera. Es común escribir los PBIs en forma de Historias de Usuario [Cohn, 2004], pero no es un requisito de Scrum. Estos PBIs son refinados por la actividad de Refinamiento de Backlog hecha por el Product Owner y el Equipo de Desarrollo mientras se practica el desarrollo de un Sprint y son priorizados por el ProductOwner. En la reunión de planificación "Planning" se toman PBIs que cumplan el criterio de aceptación o "Definition of Ready" (2 "Selection") para ser incluidos en el "Sprint Backlog" y el Equipo de Desarrollo pueda trabajar en ellos en el Sprint. A medida que el Equipo de Desarrollo termina un ítem "Sprint Backlog" cumpliendo el criterio de aceptación "Definition of Done" (3 "increment") se genera un Incremento de Producto candidato potencialmente entregable (Potentially Shippable Product Increment). Luego en la Revisión se acepta el Incremento de Producto candidato pasando a ser efectivamente un Incremento de Producto listo para ser entregado o desplegado, para "Release" (4 "releasing"). En caso de no ser aprobado (5 "Rejection") pasa nuevamente al Product Backlog para ser tenido en cuenta en el próximo Sprint. Los ítems de "Sprint Backlog" que no se lograron terminar tam-

bién vuelven (6 "comeback") al "Product Backlog". Esto ocurre formalmente en la revisión [Martin Alaimo, 2014].

## 4.5 Reglas y Consideraciones

Además se establecen algunas consideraciones relacionadas a tiempos y tamaños. Se aconseja un tamaño de equipo de desarrollo mayor a 4 miembros y menor a diez ( $7 \pm 2$ ), o sea entre cinco y nueve. Por debajo de este número mínimo se tiene un equipo pobre que puede brindar un producto pobre y por sobre ese número máximo se aumenta la complejidad de gestión y coordinación del equipo disminuyendo el funcionamiento apropiado tras la metodología planteada. También se recomienda una duración de Sprint no mayor a un mes [Ken/Jeff, 2013]. La Reunión de Planificación de Sprint debería tener un máximo de duración de ocho horas para un Sprint de un mes. El Scrum Diario o "daily" es una reunión con un bloque de tiempo de 15 minutos para que el Equipo de Desarrollo sincronice sus actividades y cree un plan para el día. Por ejemplo, una daily de mayor de 15 minutos se puede considerar larga y en 15 minutos es complicado que más de 15 personas puedan exponer lo que hicieron, lo que harán y si tienen bloqueos. Por eso se aconseja como máximo nueve personas en el equipo de desarrollo. En lo que concierne a la Revisión, el tiempo estipulado es de cuatro horas para Sprints de un mes o dos horas para uno de una quincena. La reunión de Retrospectiva debería estar restringida a un bloque de tiempo de tres horas para Sprints de un mes o a una hora y media para los de una quincena. Las reuniones de Planificación, Revisión y Retrospectiva deberían ser proporcionales a la duración del Sprint.

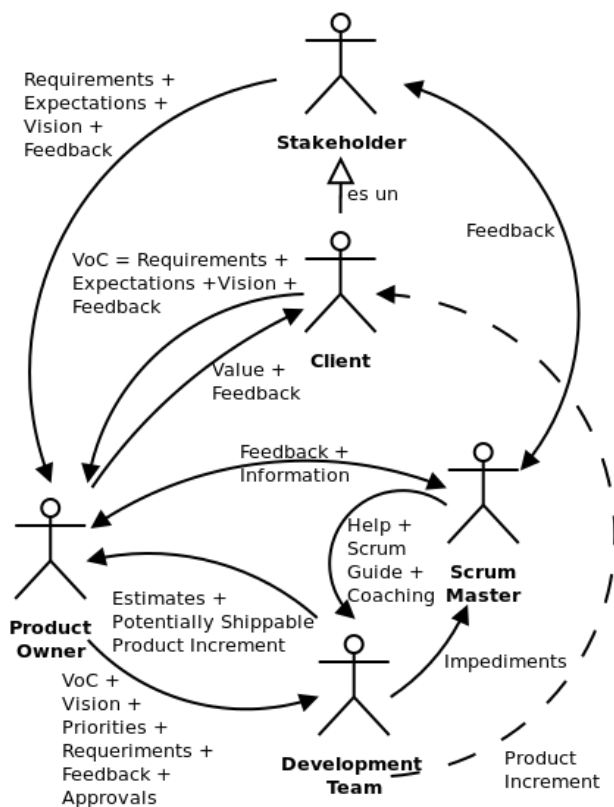


Figura 4.2: Diagrama del Sistema de Roles Scrum

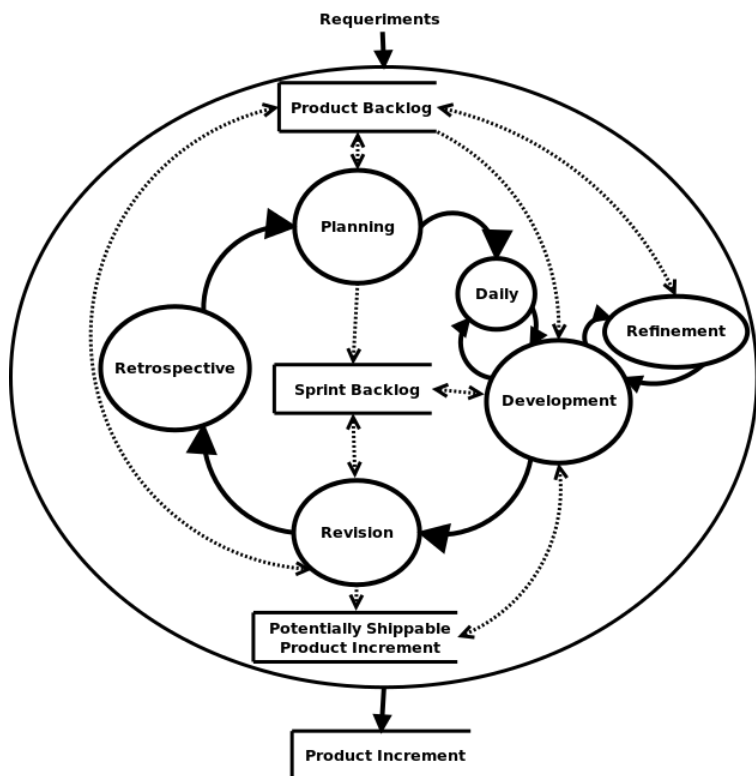


Figura 4.3: Diagrama de Flujo de Datos del Proceso Scrum

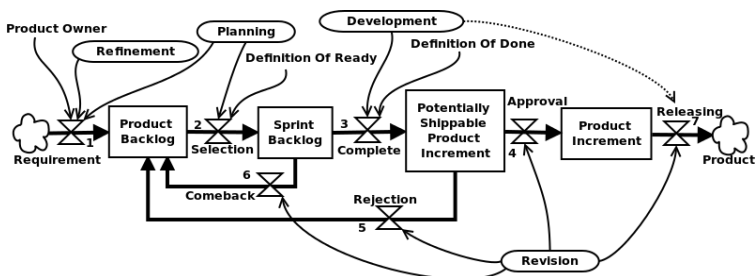


Figura 4.4: Diagrama de Flujo de Stock de artefactos Scrum

## Capítulo 5

# Gestión de Proyectos

Scrum es un marco de gestión de proyectos para el desarrollo incremental de productos, valiéndose de equipos autoorganizados.

### 5.0.1 Proyecto Scrum

Un proyecto, en ingeniería de software, es un esfuerzo temporal que se lleva a cabo para crear un sistema, software o resultado único<sup>1</sup>. Los proyectos son organizados, en una empresa u organización, por el proceso de administración de proyectos. Según este proceso, el ciclo de vida de los proyectos se puede dividir en tres fases: inicio, ejecución y cierre (ver figura 2.3). En la administración de proyectos es necesario planificar los proyectos y dicha actividad se suele hacer en la fase de inicio ("Starting phase" o "Project Start"). Pero, a diferencia de la metodología clásica (ver figura 2.3) en que la planificación estaba siempre al inicio y el desarrollo en la fase de ejecución, en el marco Scrum la planificación se distribuye durante todo el ciclo de vida del proyecto y en la fase de ejecución se hace el desarrollo incremental de productos (por incrementos de producto) en iteraciones cortas (Sprint) donde cada iteración tiene su respectiva planificación (ver figura 5.1) y su incremento de producto, en caso de haberlo logrado.

---

<sup>1</sup>Se parafrasea la definición del PMBOOK [PMBOK, 2004].

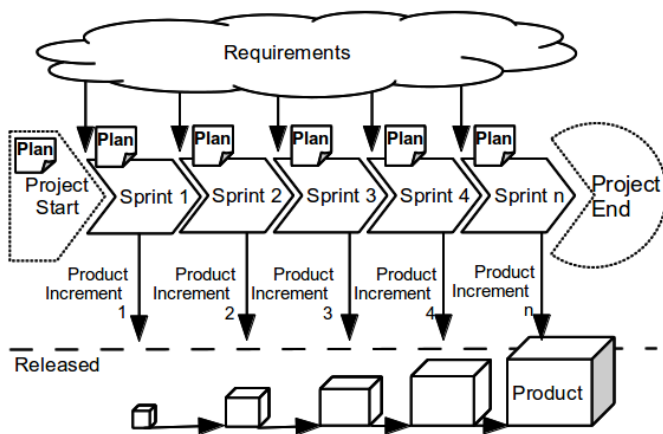


Figura 5.1: Proyecto Scrum

Entonces podemos decir que en Scrum se piensa en muchos planes periódicos (a corto plazo). Los mismo pueden estar en un plan mayor a largo plazo pero de carácter flexible. También se puede realizar un plan global de entregables en base a los incrementos de producto estimado. Pero, desde esta perspectiva, hay que considerar que aunque se trabaje con planificaciones, los planes no son contratos a respetar a rajatabla.

### 5.0.2 Triángulo de la Gestión de proyectos

El marco de Scrum cambia el triángulo clásico de la gestión de proyectos. El compromiso ya no es entre el tiempo, presupuesto y calidad; sino que se basa en el triángulo de: Presupuesto, Tiempo y funcionalidad (ver figura 5.2).

### 5.0.3 Planificación de entregables

Con esta metodología tampoco es necesario hacer una entrega final (o "releasing") ya que se pueden hacer entregas paulatinas. Para hacer entregas intermedias se puede crear un plan de muy alto

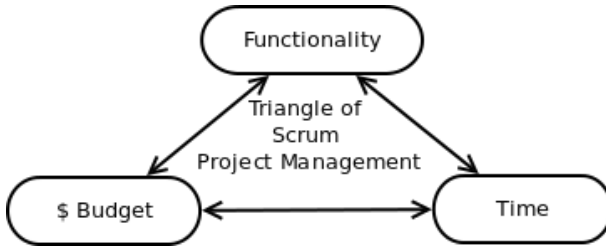


Figura 5.2: Triángulo de Gestión de Proyectos Scrum

nivel para múltiples Sprints durante una planificación de lanzamiento. Este plan de entregables o de lanzamientos es una guía con la que se pretende reflejar las expectativas sobre las qué funciones se implementarán y cuando se completarán [Scrum-Institute, 2015]. También sirve como una base para monitorear el progreso dentro del proyecto. Pero siempre hay que considerar que no es un plan equivalente a un plan clásico, los ítos de releases no deberían ser compromisos rígidos y contractuales, y el desarrollo del proyecto no debería centrarse en respetar el plan. Por este motivo el plan de lanzamiento no es un plan estático. Pues, se cambia durante todo el proyecto cuando nuevos requerimientos o conocimientos están disponible y, por ejemplo, cuando entradas en el Scrum Product Backlog cambian y re-estiman. Por lo tanto este plan debe ser revisado y actualizado en intervalos regulares, por ejemplo, después de cada Sprint.

Para crear un plan de entregables se deben tener disponibles las siguientes cosas:

1. Un Product Backlog priorizado y estimado.
2. La velocidad estimada del Equipo Scrum.
3. Las condiciones de satisfacción (metas para la agenda, el alcance, los recursos)

#### 5.0.4 Medición y métricas

Todo...





# Capítulo 6

## Escalamiento

Scrum es aconsejable para ser óptimo en equipos chicos y proyectos pequeños, con agrupación de personas de múltiples disciplinas en un solo equipo para maximizar el ancho de banda de las comunicaciones, la visibilidad y la confianza. Esto sucede porque cuando los equipos son grandes aumenta el acoplamiento de individuos complejizando las comunicaciones y dificultando la coordinación y el buen desarrollo de las reuniones Scrum. Además se desprende del principio o Ley de Brooks , que cuando se agregan personas a un proyecto o equipo aumentan los canales de comunicación pudiendo generar sobrecarga de comunicación. Por este motivo y desde un punto de vista purista, cuando se quiere implementar Scrum en proyectos grandes que requieren muchas personas, en su forma ortodoxa, no es recomendable. Pero se han encontrado maneras organizativas para aplicar Scrum en estos casos. Una forma es el uso de equipos Scrum de características, equipos Scrum de componentes y el uso de "Scrum de Scrum".

### 6.1 Equipos de características

Abordar el problema de proyectos grandes con "Equipos Scrum de características" consiste en la conformación de equipos totalmente multi-funcionales, capaces de operar en todos los niveles de

la arquitectura del producto con el fin de ofrecer las características centradas en el cliente. O sea que cada equipos trabaja sobre determinadas características de producto (Features) o PBIs desarrollando todos los niveles del sistema a desarrollar. En este sentidos, los equipos son homogéneos entre si pero eterogéneos internamente, con integrantes de habilidades diversas y características profesionales multidisciplinares. Para lograr esto se debe conformar una organización de aprendizaje donde los equipos practican el aprendizaje continuo, donde aprenden para abarcar los componentes arquitectónicos.

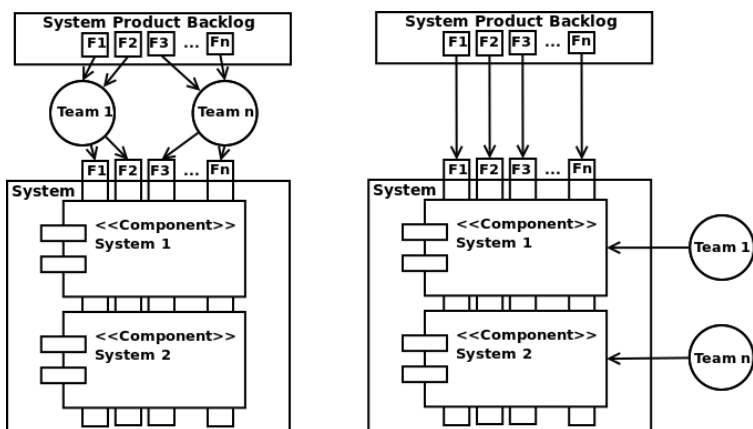


Figura 6.1: Equipos de Características y Equipos de Componentes.

Equipos de Características (izquierda) y Equipos de Componentes (derecha)

En esta arquitectura de organización es necesario coordinar el trabajo de los diferentes equipos. Los Scrum Master deben reunirse con regularidad, promoviendo la transformación a travez de una lista visible de los impedimentos de organización. Los Scrum Master, además deberan estar familiarizados con biografia relacionada a este proble de escalabilidad como "Scaling Lean and Agile Development" [Larman/Vodde, 2008].

## 6.2 Equipos de componentes

Abordar el problema de proyectos grandes con "Equipos Scrum de componentes" (ver figura 6.1) consiste en que cada equipo sólo es responsable de la ejecución de ciertos componentes dedicados en el sistema de los cuales el equipo es dueño de su desarrollo. Desde esta perspectiva se pueden tener equipos dedicados por capas (capa front-end, capa de servicios, capa de persistencia) y por componentes de arquitectura de software (diferentes componentes como librerías, servicios o subsistemas).

Para terminar una PBI o historia de usuario hay en la mayoría de los casos la necesidad de dividir las historias en partes más pequeñas que podrían ser implementadas dentro de un solo componente. Además se genera dependencias entre los diferentes equipos haciendo necesario procesos de integración periódica y coordinación de equipos. En muchos casos, una sola historia de usuario no se puede implementar dentro de un único sprint y, en su defecto, depende de los resultados de otras historias desarrolladas por otro equipo que aún no están disponibles. A esto se lo llama "pipeline" y debe evitarse en lo posible o gestionarse apropiadamente.

La ventaja de utilizar equipos de componentes es que es más fácil asegurar una determinada arquitectura del sistema. Por ejemplo si se quiere asegurar una Arquitectura SOA o una de Microservicios que está orientada a componentes. Esta idea está, entre otras cosas, basada en la "Ley de Conway"<sup>1</sup> que sugiere que las organizaciones pueden replicar su arquitectura en los productos que ellas producen [Martin Fowler, 2014].

Por otro lado, puede tener como desventaja que las personas se pueden especializar sólo en pequeñas partes del sistema y el conocimiento global sobre el sistema en su conjunto podría perderse [Scrum-Institute, 2015]. En este caso podría tener lugar una optimización local, ya que el equipo a veces puede tomar decisiones que están optimizadas para el componente individual, pero las mejores soluciones desde una perspectiva del sistema total podrían haber sido desestimadas u oviadas.

---

<sup>1</sup>Conway's Law [Conway, 1968] no es exactamente una ley, sino que es más bien una observación que Conway publicó en 1968.

## 6.3 Equipos mixtos

También es posible la configuración de equipos mixtos que son la combinación de equipos de características y de componentes (ver figura 6.2).

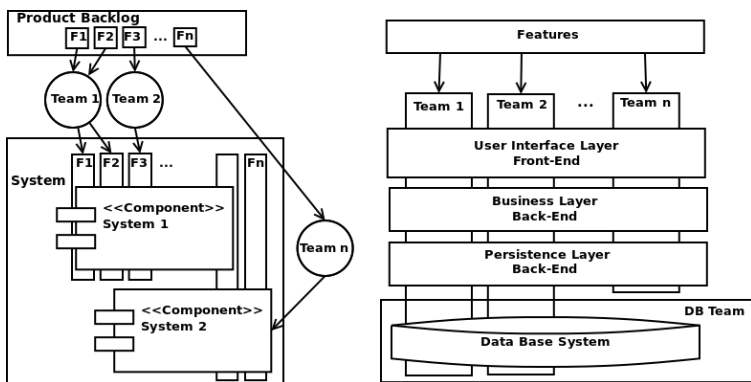


Figura 6.2: Ejemplo de equipos mixtos.

## 6.4 Scrum de Scrum

Scrum de Scrum es una forma de organización y técnica para escalar Scrum a grupos grandes de personas. Consiste en distinguir un integrante con el rol de Embajador, denominado "Embajador", por cada Equipo Scrum [Stefanini, 2013]. El Embajador será quien participará en reuniones Daily con Embajadores de otros equipos. A esta reunión de Embajadores se la llama "Scrum de Scrum". Habitualmente se usa que el rol de Embajador lo desempeñe el Scrum Master, pero puede ser desempeñado por otro integrante del Equipo de Desarrollo. La reunión "Scrum de Scrum" se comporta como una Daily donde los Embajadores reportan la situación de su equipo y sus impedimentos.

## Capítulo 7

# Complementos de Scrum

Rodeando el núcleo de Scrum se encuentra un anillo de complementos. Estos complementos son métodos y técnicas complementarias a Scrum, sugerencias y recomendaciones, lecciones aprendidas y conceptos aledaños. En este capítulo trataremos estos temas.

## 7.1 Técnicas de Comunicación

### 7.1.1 Gestión visual

La gestión visual es la utilización de elementos y técnicas visuales, como complemento de Scrum, para la organización del trabajo y la irradiación visual del trabajo. Es aconsejable que los irradiadores de información presenten las siguientes características:

1. **Ubicación ostensible:** estar ubicado en el lugar de trabajo en forma de afiches, carteles o pizarras visibles.
2. **Soporte físico:** estar hechos de material tangible como papel en vez de usar software, salvo el caso de monitores grandes.
3. **Resumen autoexplicativo:** contener información importante autoexplicativa y didáctica.

Ejemplos de irradiadores de información son: el gráfico burn-down, indicadores de estados del build (usando semáforos), métricas de errores, riesgos activos, impedimentos. etcétera.

### 7.1.2 Tableros Scrum/Kanban

El tablero Scrum/Kanban o "Scrum Kanban Board" es una técnica que consiste en utilizar un tablero Kanban (ver figura 7.1), como irradiador de información, para el manejo del ciclo de estados de las tareas y de los impedimentos. El tablero Kanban debe ser visible por todo el equipo y por lo tanto transparente para todos los involucrados. Por ejemplo, durante una Daily Scrum todo el equipo es capaz de ver qué tareas se resuelven, cuáles no se han abordado todavía y qué impedimentos existen.

#### Ejemplo de un Scrum kanban board

Ver figura 7.1.











PB	SB	TO DO	ON GOING	DONE	BLOCKED	OK
						
						
						
						

Figura 7.1: Tablero Kanban para Scrum

**Ejemplo de un Scrum board**

Un Scrum board, Kamban Board o taskboard físico (Ver figura 7.2) es una tabla donde se colocan postick representando tareas.

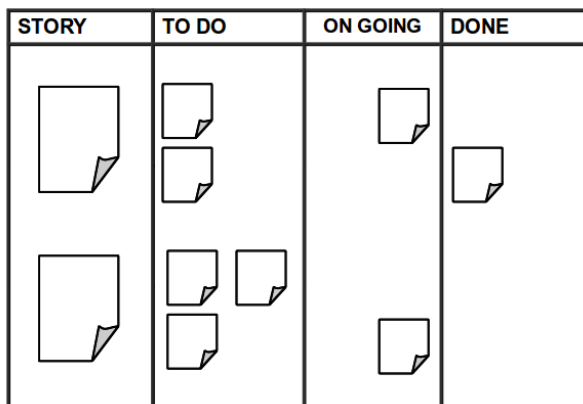


Figura 7.2: Tablero Scrum

## 7.2 Malas prácticas

Existen muchas causas diversas de que se fracase en la implementación de Scrum. A esas acciones causadas resultado de prácticas y que perjudican el buen funcionamiento de Scrum las podemos llamar malas prácticas y constituirán prácticas que se aconseja evitar. A continuación se listan algunas de ellas:

### 1. Aplicar mal la metodología

Una mala práctica es aplicar mal o en forma incompleta una metodología o técnica determinada. Por ejemplo, se critica a la metodología de cascada (Waterfall) o desarrollo en cascada porque se dice que no funciona, sin embargo lo que suele suceder es que no se aplica realmente (Masa Maeda <sup>1</sup>).

El problema no es que no sirva o no funcione, sino que no lo

---

<sup>1</sup>Masa K Maeda es PhD, Founder and CEO de Valueinnova USA (capacitación Scrum). Es el creador de Serious LeAP, consultor senior del Cutter Consortium en Boston, miembro del comité de dirección del Agile Testing Alliance y maestro en la Universidad de California en Berkeley. Es pionero de Lean y Kanban para trabajo de conocimiento y uno de los formadores del Lean Kanban University. Es una figura líder mundial en Agile y ha generado Serious Games de alto calibre.



hacemos bien (Masa Maeda 2012). Pues, en el artículo original de 1970 en el que Royce Winston expone el desarrollo en cascada se habla de ciclos y de “fases sucesivas de desarrollo iterativo” [Winston Royce, 1970] ofreciendo unos consejos a seguir, cosa que no se suele hacer y se da por hecho que cascada no sirve. Algo parecido ocurre con Scrum.

Hay quienes dicen: "Scrum fracasó". Pero, sin embargo, lo que suele suceder es que se dice que se implementa Scrum aunque, en la práctica, no se cumplen sus recomendaciones o se hacen hibridaciones con otras metodologías que dan como resultado algo que no es Scrum. Creyendo hacer Scrum, muchas no han logrado superarlo [Gantthead-James, 2010].

Respecto a este tema en una entrevista que le hicieron a Jeff Sutherland (uno de los creadores de Scrum) él dijo: La mayoría de las empresas implementan Scrum a medias. Por ejemplo, cualquier Scrum sin producto de trabajo al final de un Sprint es un Scrum fracasado y el 80 por ciento del Scrum escalado en Silicon Valley se encuentra en esta categoría, pues son "ágiles sólo de nombre".

## 2. Aplicar mal un principio

En ocasiones en que se aplica mal una metodología se puede deber a mal interpretar sus principios rectores, a generalizarlos o a caer en un una especie de fundamentalismo. Por ejemplo, si consideramos el "trabajo empírico" como algo fundamental, pero basándonos en ello pretendemos que todo trabajo se base en la experiencia directa de los desarrolladores (tipo prueba error) sin recurrir a experiencias previas, a memoria histórica, a procesos organizacionales o a conocimiento teórico. De este modo se puede caer en ser un equipo de trabajo reactivo, ciego de las consecuencias a largo plazo de sus acciones e ineficiente. No es necesario reinventar la rueda cada vez que se nos presenta la necesidad de usar una y a veces eso es lo que ocurre cuando se abusa del trabajo basado en la prueba y el error. Por otro lado, cuando nuestros actos tienen consecuencias que trascienden el horizonte de aprendizaje (nuestra experien-

cia cercana), se vuelve imposible aprender de la experiencia directa [Peter Senge, 1990]. En ocasiones, por poner otro ejemplo, se hace énfasis en el coraje. Pero se puede caer en ser heroico, cuando como dice una frase: "el programador heroico a menudo no ve el gran dragón" (Software Architect Bootcamp). Pues fomentar el coraje no es fomentar necesariamente las acciones individuales heroicas en vez de buscar sentirse apoyados y tener más recursos a disposición para promover el coraje para enfrentar desafíos más grandes.

### 3. No hacer ingeniería por aplicar una metodología

Metodologías como Scrum no establecen las prácticas específicas de ingeniería, por lo que se puede aplicar la metodología sin hacer ingeniería. En este caso los ScrumMasters son responsables de promover un mayor rigor de la aplicación de las practicas ingenieriles y de la definición de "terminado" (DoD) acorde al marco de ingeniería [Ganttthead-James, 2010].

### 4. No se cambia de mentalidad

A veces se implementa y usa una nueva metodología pero no se cambia de forma de pensar. Es que se puede considerar que la simple adopción de una herramienta de trabajo, sin una transformación personal que la acompañe, no sirve de nada, por ejemplo adoptar Scrum sin una transformación personal [Martin Alaimo/Kleer, 2014].

La gran mayoría de metodologías tienen detrás principios y maneras de pensar. Pues hay que entender que no se trata solo de fórmulas, sino también de formas de razonar. No se puede pretender trabajar en un equipo con alguna metodología ágil que implementa auto-organización si no se cree en la auto-organización. Hay personas que creen en el liderazgo centralizado y autoritario y no cambian su perspectiva y en vez de adaptarse a la nueva manera de trabajar terminan queriendo adaptar la manera de trabajar a su idea original, por ejemplo a la conducción centralizada y autoritaria. Esta actitud termina por generar malas prácticas que socaban el buen funcionamiento de una metodología determinada.

### 5. **Disociación entre producción y resto de la organización**

La disociación entre producción y resto de la organización se da cuando se implementa Scrum solo en el área de producción para hacer Scrum en el proceso de desarrollo, pero la gestión estratégica o las capas de gestión de alto nivel de la compañía desconocen la filosofía Scrum y gestionan los portfolios, programas y proyectos con las metodologías criticadas por Scrum. Algo semejante sucede con los vendedores de la organización. Pues, si los mismos venden productos especificados de antemano y en base a esas ventas se realizan compromisos contractuales rígidos y se exige que el área de producción cumpla con esos compromisos, por más que el área de producción intente usar Scrum se puede caer en los mismos problemas que Scrum critica e incumplir con el o los proyectos.



## Capítulo 8

# Glosario y Acrónimos

Nombre	Descripción
DoD	Definición de terminado (Definition of Done)
DoR	Definición de completitud (Definition of ready)
DT	Equipo de desarrollo (Development Team)
PB	Backlog de producto (Product Backlog)
PI	Incremento de producto (Product Increment)
PMI	Instituto de Gestión de Proyectos (Project Management Institute)
PO	Dueño del producto (Product Owner)
ROI	Retorno de la inversión (Return On Investment)
SCRUM	Es un juego de rugby en el que, por lo general, tres miembros de cada línea se unen opuestos unos a otros con un grupo de dos y un grupo de tres jugadores detrás de ellos, lo que hace un grupo de ocho personas, tres, dos, tres formados en cada lado; el balón se deja entre la línea divisoria de ambos grupos, los jugadores están abrazados y tomados de la cintura de un compañero de equipo y los del frente hombro a hombro con el oponente, y se trata hacer fuerza grupalmente para desplazar al grupo rival y patear la pelota hacia atrás para que un compañero de equipo la tome.
SB	Backlog de iteración (Sprint Backlog)
SM	Facilitador (ScrumMaster)
SP	Iteración (Sprint)

# Bibliografía

- [Agile Atlas, 2012] Agile Atlas (2012). *Agile Atlas. Scrum, una descripción.* by Scrum Alliance. Scrum Alliance Core Scrum 2012-2013.
- [Anacleto, 2005] Anacleto (2005). *El rol de la arquitectura de software en las metodologías ágiles.* por Lic. Valerio Adrián Anacleto. Epidata Consulting S.R.L.- Buenos Aires, Argentina. Diciembre de 2005.
- [AntiPatterns, 1998] AntiPatterns (1998). *AntiPatterns Refactoring Software, Architectures, and Projects in Crisis.* By William J. Brown Raphael C. Malveau Hays W. McCormick III Thomas J. Mowbray John Wiley and Sons. Inc. Publisher: Robert Ipsen, 1998.
- [Austin, 2003] Austin (2003). *Artful Making: What manager need to know about how artist work.* By Austin Robert D., Devin, Lee. Prentice Hall 2003.
- [Beck, 2001] Beck (2001). *Agile Manifesto By Beck, Kent.* URL: [www.agilemanifesto.org](http://www.agilemanifesto.org), 2001, como estaba en Octubre de 2012.
- [CHAOS Report, 1994] CHAOS Report (1994). *The CHAOS Report.* By Standish Group. Standish Group.
- [Cohn, 2004] Cohn (2004). *User Stories Applied: For Agile Software Development.* By Mike Cohn. Addison Wesley.
- [Conway, 1968] Conway (1968). *Article: How Do Committees Invent?* By Melvin E. Conway. Copyright 1968, F. D. Thompson

Publications, Inc. Reprinted by permission of Datamation magazine, where it appeared April.

[Eric Raymond, 1997] Eric Raymond (1997). *The Cathedral and the Bazaar*. By Eric S. Raymond. O'Reilly.

[Ganttthead-James, 2010] Ganttthead-James (2010). *Seven Obstacles to Enterprise Agility*. By Ganttthead, James. ganttthead.com.

[James Surowiecki, 2005] James Surowiecki (2005). *The Wisdom of Crowds*. By James Surowiecki. Anchorsbooks, August 16, 2005.

[Ken Schwaber, 1995] Ken Schwaber (1995). *SCRUM Development Process* by Ken Schwaber. Proceedings of the 10th Annual ACM Conference on Object Oriented Programming Systems, Languages, and Applications OOPSLA.

[Ken Schwaber, 2002] Ken Schwaber (2002). *Agile software development with Scrum* By Ken Schwaber. Prentice Hall. ISBN 0-13-067634-9.

[Ken Schwaber, 2011] Ken Schwaber (2011). *Agility and PMI* By Ken Schwaber. KenSchwaber.wordpress.com.

[Ken/Jeff, 2013] Ken/Jeff (2013). *La Guía de Scrum. La Guía Definitiva de Scrum: Las Reglas del Juego*. por Ken Schwaber y Jeff Sutherland. Offered for license under the Attribution Share-Alike license of Creative Common.

[Larman/Vodde, 2008] Larman/Vodde (2008). *Scaling Lean and Agile Development: Thinking and Organizational Tools for Large-Scale Scrum*. By Craig Larman, Bas Vodde. Paperback.

[Lawson/Martin, 2008] Lawson/Martin (2008). *On the Use of Concepts and Principles for Improving Systems Engineering Practice*. By Lawson, H. and J. Martin. INCOSE International Symposium 2008, 15-19 June 2008, The Netherlands.



- [Martin Alaimo, 2014] Martin Alaimo (2014). *Proyectos Ágiles con Scrum. Flexibilidad, apredizaje, innovación y colaboración en contextos complejos. Por Martin Alaimo.* Kleer (Agile Coaching and training).
- [Martin Alaimo/Kleer, 2014] Martin Alaimo/Kleer (2014). *Equipos más productivos: Personas e interacciones por sobre procesos y herramientas. Por Martin Diego Alaimo.* Kleer (Agile Coaching and training), Buenos Aires. ISBN 978-987-45158-7-2.
- [Martin Fowler, 2014] Martin Fowler (2014). *Article: Microservices. By Martin Fowler, James Lewis.* Martin fowler blog. Article 25 March 2014.
- [MIT Press, 2009] MIT Press (2009). *The Wisdom of Crowds in the Recollection of Order Information. By Steyvers, M., Lee, M.D., Miller, B., and Hemmer, P. (2009). In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams and A. Culotta (Eds.) Advances in Neural Information Processing Systems, MIT Press, 2009.*
- [Neal Ford, 2010] Neal Ford (2010). *Arquitectura evolutiva y diseño emergente: Investigación sobre arquitectura y diseño. By Neal Ford, Application Architect. IBM, ThoughtWorks Inc., DeveloperWorks, 05-04-2010.*
- [Peter Senge, 1990] Peter Senge (1990). *La quinta disciplina. El arte y la práctica de la organización abierta al aprendizaje. By Peter M. Senge.* Editorial Granica, 2003. Edición original en inglés, 1990.
- [PMBOK, 1996] PMBOK (1996). *A Guide to the Project Management Body of Knowledge (PMBOK Guide). By PMI and William R. Duncan Director of Standards.* PMI Standards Committee.
- [PMBOK, 2004] PMBOK (2004). *Guía de los Fundamentos de la Dirección de Proyectos Tercera Edición (Guía del PMBOK).* Project Management Institute, Four Campus Boulevard, Newtown Square, PA 19073-3299 EE.UU.

- [SBOK, 2013] SBOK (2013). *Una guía para el conocimiento de Scrum (Guía SBOK) - 2013 Edición. Título original: A Guide to the SCRUM BODY OF KNOWLEDGE (SBOK GUIDE) 2013 Edition*. SCRUMstudy, una marca de VMEdU, Inc.
- [Scrum Alliance, 2015] Scrum Alliance (2015). *Scrum Alliance* ([scrumalliance.org](http://scrumalliance.org)). Scrum-Alliance.
- [Scrum-Institute, 2015] Scrum-Institute (2015). *Scrum revealed: the only book can simply learn scrum! by International Scrum Institute*. [scrum-institute.org](http://scrum-institute.org) o ISI.
- [ScrumManager, 2014] ScrumManager (2014). *Gestión de proyectos Scrum Manager (Scrum Manager I y II) By Juan Palacio*. De la edición: Scrum Manager (Creative Commons Attribution Non-Commercial 3.0).
- [SEBoK, 2014] SEBoK (2014). *Guide to the Systems Engineering Body of Knowledge (SEBoK) v. 1.3. By Body of Knowledge and Curriculum to Advance Systems Engineering (BKCASE) project*. [Sebokwiki.org](http://Sebokwiki.org), released 30 May 2014.
- [Sriram Narayan, 2015] Sriram Narayan (2015). *Agile IT Organization Design: For Digital Transformation and Continuous Delivery. por Sriram Narayan*. Addison-Wesley Professional. Release Date: June 2015. ISBN: 9780133903690.
- [Stefanini, 2013] Stefanini (2013). *Scrum of Scrums: Running Agile on Large Projects. By Leandro Faria Stefanini*. Scrum Alliance, 5 June 2013.
- [Steven Johnson, 2002] Steven Johnson (2002). *Emergence: The Connected Lives of Ants, Brains, Cities, and Software Emergence. By Steven Johnson*. Prentice Hall 2002.
- [Takeuchi/Nonaka, 1986] Takeuchi/Nonaka (1986). *The New New Product Development Game. by Hirotaka Takeuchi, Ikujiro Nonaka*. Harvard Business Review.
- [UNTREF, 2014] UNTREF (2014). *Construcción de software: una mirada ágil. Por Nicolás Paez, Diego Fontdevila,*

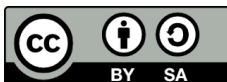
*Pablo Suárez, Carlos Fontela, Marcio Degiovannini, Alejandro Molina.* Universidad Nacional de Tres de Febrero (UNTREF).

[Wiki, 2015] Wiki (2015). *Wikipedia*, 2015. Online es.wikipedia.org.

[Wiley/Sons, 2002] Wiley/Sons (2002). *Agile Modeling: Effective Practices for Extreme Programming and the Unified Process.* By John Wiley and Sons. Paperback – March 21, 2002. ISBN: 0471202827.

[Winston Royce, 1970] Winston Royce (1970). *Managing the Development of large Software Systems by Dr. Winston W. Royce.* Proceedings of IEEE WESCON 26 (August).

Los artículos y las ilustraciones de este libro se distribuyen bajo una licencia Creative Commons by-sa 4.0



[http://creativecommons.org/licenses/by-sa/4.0/deed.es\\_AR](http://creativecommons.org/licenses/by-sa/4.0/deed.es_AR)

Pueden ser copiados, distribuidos y modificados bajo las condiciones de reconocer a los autores y mantener esta licencia para las obras derivadas.