Tema 2

Scrum

Ingeniería del Software Avanzada

Índice

[Esquema 3](#_Toc535052607)

[Ideas clave 4](#_Toc535052608)

[2.1. Introducción y objetivos 4](#_Toc535052609)

[2.2. Orígenes y filosofía de Scrum 4](#_Toc535052610)

[2.3. Valores de Scrum 6](#_Toc535052611)

[2.4. Flujo del proceso Scrum 8](#_Toc535052612)

[2.5. Los roles de Scrum 9](#_Toc535052613)

[2.6. Los artefactos de Scrum 12](#_Toc535052614)

[2.7. Los eventos de Scrum 14](#_Toc535052615)

[2.8. Las historias de usuario 18](#_Toc535052616)

[2.9. Extensiones de Scrum 21](#_Toc535052617)

[2.10. Estimaciones ágiles 24](#_Toc535052618)

[2.11. Escalado de Scrum 27](#_Toc535052619)

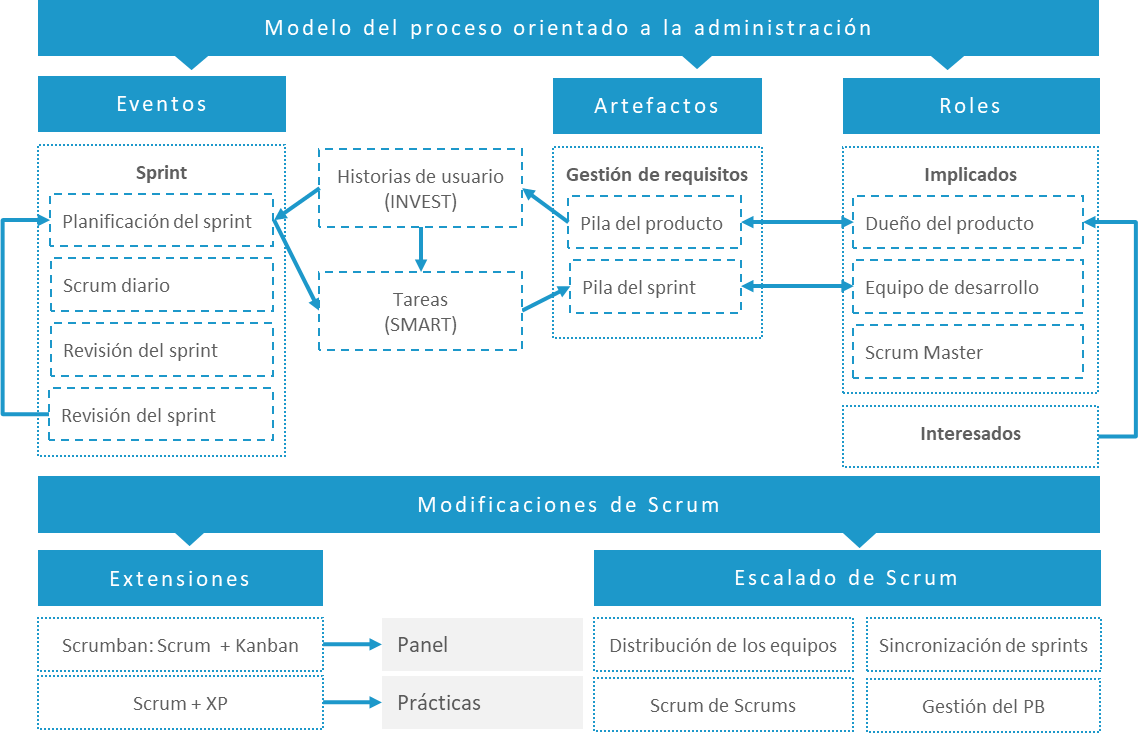
[2.11. Referencias bibliográficas 33](#_Toc535052620)

[A fondo 35](#_Toc535052621)

[Actividades 38](#_Toc535052622)

[Test 42](#_Toc535052623)

Esquema



Ideas clave

2.1. Introducción y objetivos

En este tema vamos a estudiar a fondo **Scrum, una de las metodologías ágiles más populares, simples y extendidas**. Como el resto de este tipo de metodologías, sigue los **principios propuestos** en el **«Manifiesto ágil»**, y los concreta en una serie de indicaciones metodológicas relacionadas con la definición de roles específicos, reuniones y artefactos, que soportan el proceso de desarrollo y permiten guiar el proyecto utilizando una **documentación muy ligera a la vez que efectiva**.

Con el estudio de este tema pretendemos alcanzar los siguientes **objetivos**:

* Conocer el **enfoque general** del proceso de desarrollo bajo Scrum y su relación con los principios del «Manifiesto ágil».
* Conocer los **principales roles** que intervienen en el proceso, su implicación, y el papel que juegan en cada fase del desarrollo.
* Conocer las **principales reuniones propuestas por Scrum**, y sus beneficios desde el punto de vista del avance del proyecto y el intercambio de información.
* Comprender y saber utilizar los **principales artefactos**, pensados para mantener la especificación de requisitos y controlar el avance del proyecto en su conjunto.
* Saber **extender la metodología básica** de Scrum, pensada inicialmente para equipos de tamaños reducido, a **proyectos de mayores dimensiones**.

2.2. Orígenes y filosofía de Scrum

Scrum es una **metodología de desarrollo de software** desarrollada por Jeff Sutherland a comienzos de la década de 1990. El nombre proviene de una formación propia del ***rugby***, en la que dos grupos de jugadores, de equipos diferentes, se colocan alrededor del balón y en oposición violenta para ganar el control de este y desplazarlo alrededor del campo.

Scrum no es propiamente una metodología, sino más bien un **marco general de trabajo** para el desarrollo de productos complejos, que se centra en la **administración iterativa del proceso de desarrollo**,   
y no tanto en prácticas específicas.

No prescribe actividades concretas ni ofrece una especificación detallada de la manera en que deben organizarse las tareas de desarrollo, sino que **proporciona un contexto** en el que puedan ser los propios involucrados quienes creen su propio proceso. En este sentido, como veremos, **Scrum es especialmente apropiada para combinarla con prácticas y herramientas procedentes de otras metodologías**.

Scrum tiene una **fundamentación eminentemente empírica**, defendiendo que cualquier decisión debe tomarse a partir del conocimiento acumulado por la experiencia.

Para poder establecer un control del proceso empírico, Scrum se asienta sobre **tres pilares** (figura 1) (Schwaber y Sutherland, 2017):

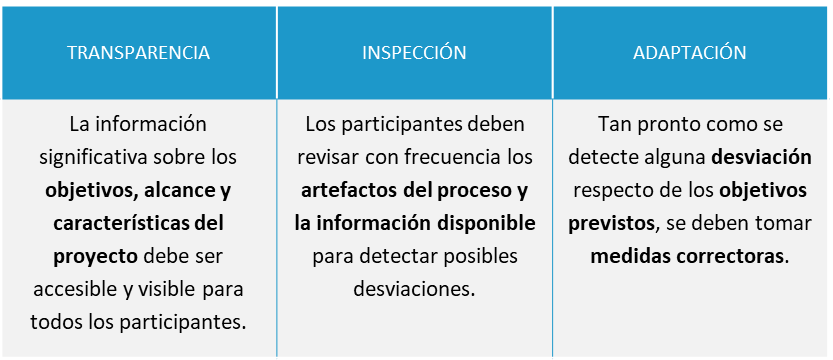


Figura 1. Pilares de Scrum

2.3. Valores de Scrum

Scrum se adhiere a los cuatro valores del «Manifiesto ágil». Así, en la figura 2, son contemplados por esta metodología (Alaimo, 2013).

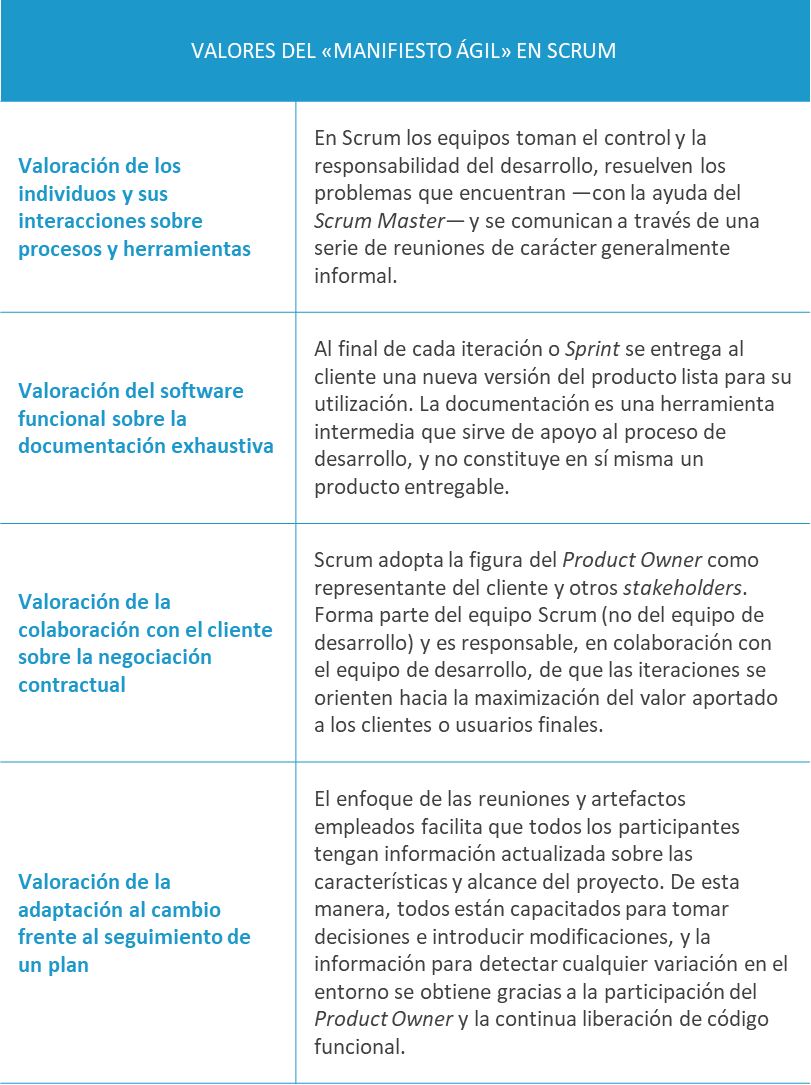


Figura 2. Valores generales de Scrum

Estos valores generales son acompañados y reforzados por una serie de **valores sobre los que se apoya la metodología** (Alaimo, 2013; Schwaber y Sutherland, 2017), como se ve en la figura 3:

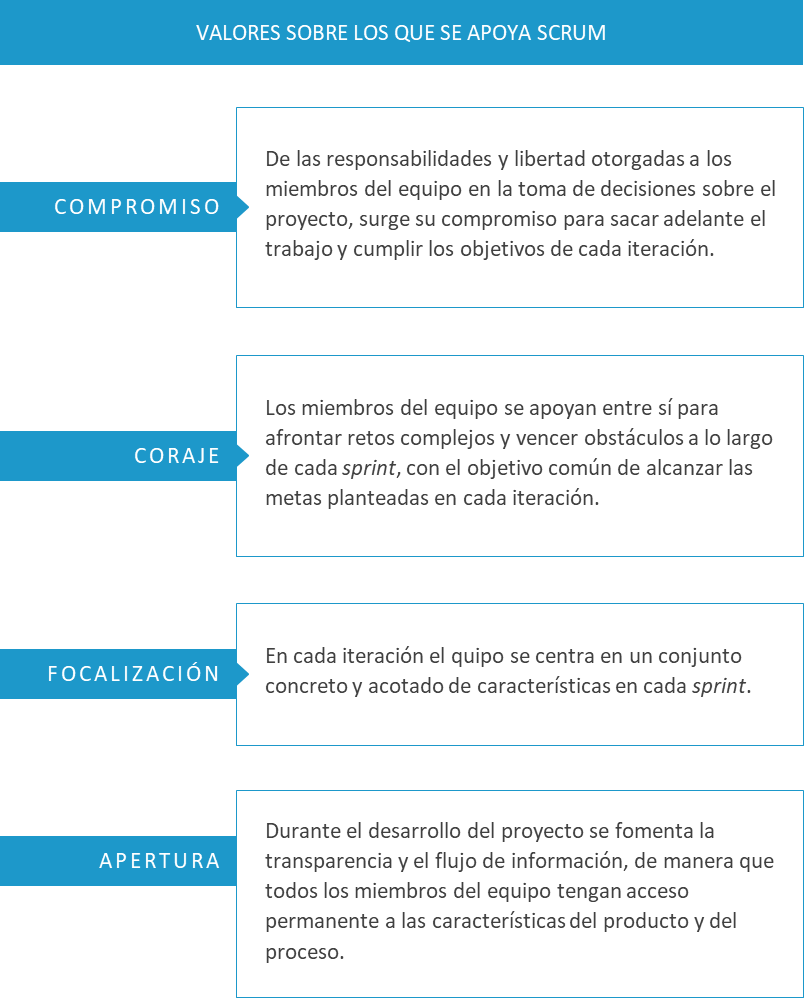


Figura 3. Valores de Scrum

2.4. Flujo del proceso Scrum

En la figura 4 se muestra un **esquema general del modelo de proceso** que adopta Scrum.

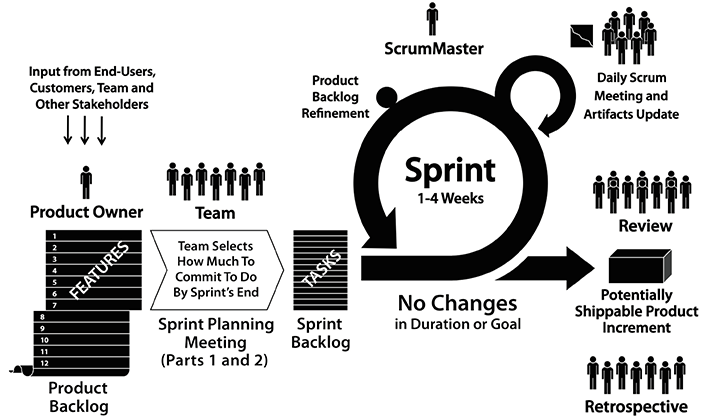


Figura 4. Modelo de proceso de Scrum, incluyendo roles, artefactos y eventos.

Fuente: Sutherland y Schwaber (2012)

En este esquema se muestran los **principales roles** (*Scrum Master*, *Product Owner* y Equipo), **reuniones** (Planificación de *sprint*, Scrum diario, revisión y retrospectiva del *sprint*) y **artefactos** (la pila del producto y la pila del *sprint*).

Lo fundamental es comprender que se trata de un **proceso iterativo con comunicación y retroalimentación constantes**. La definición de la «pila del *sprint*» durante la reunión de «planificación» marca el comienzo de cada iteración, y el equipo de desarrollo se compromete con el *Product Owner* en la entrega de un conjunto de funcionalidades identificadas en este documento.

Durante cada *sprint*, que son períodos de duración fija, se realiza al comienzo de cada jornada una **reunión** donde los miembros del equipo de desarrollo comparten información sobre el progreso de la iteración. Al final de cada *sprint* se muestra al *Product Owner* el incremento realizado, y el equipo reflexiona sobre la práctica realizada, identificando posibilidades de mejora sobre el proceso.

A lo largo de las próximas secciones iremos analizando en profundidad cada uno de estos elementos, y su encaje en el esquema global.

2.5. Los roles de Scrum

Los **equipos de Scrum son autorganizados y multifuncionales** y están compuestos por tres roles principales (Schwaber y Sutherland, 2017):

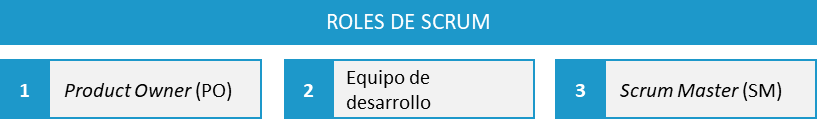


Figura 5. Roles principales de Scrum

El *Product Owner* (PO)

Es el **responsable último del éxito del producto**, que debe ajustarse a las necesidades de cliente y usuarios finales. Tiene la visión general del producto, define los requisitos y comprende los puntos de vista del cliente y otros *stakeholders*. Su principal misión es maximizar la rentabilidad del producto o retorno de la inversión —*Return of Investment* (ROI)— y para ello debe considerar en primer lugar la priorización de las funcionalidades deseadas.

Sus **principales responsabilidades** son las siguientes:

* **Comprende los objetivos del producto** y se encarga demaximizar el valor aportado en el desarrollo en cada iteración.
* **Se** **comunica con el cliente, usuarios y *stakeholders*** para obtener información y representar sus intereses en el proyecto.
* **Se comunica con el equipo de desarrollo** para asegurarse de que todos comprenden los requisitos con el detalle suficiente.
* **Define los requisitos del producto y mantiene la pila de producto**, que va adaptando en cuanto a número de requisitos y prioridades relativas según avanza el proyecto. Para ello cuenta con la ayuda del *Scrum Master.*
* **Determina el alcance de cada iteración** en colaboración con el equipo de desarrollo, priorizando los requisitos según una visión de negocio.
* **Acepta (o rechaza) la versión entregada al final de cada *sprint*** y suministra *feedback* al resto del equipo.

El equipo de desarrollo

Es el **conjunto de profesionales que se responsabiliza de cumplir los objetivos de desarrollo** planteados en cada iteración, y de entregar al final de cada *sprint* una versión funcional que pueda ser mostrada en el ***Sprint Review* (SR)**.

Son preferiblemente de tamaño reducido, nunca más de nueve personas, y deben tener las siguientes **características**:

* Ser **autorganizados**, y solo ellos, sin intervención de ningún otro rol, determinan la manera de planificar el proceso de desarrollo en cada iteración y la descomposición de funcionalidades en tareas o su asignación a desarrolladores concretos.
* Ser **multifuncionales**, englobando perfiles heterogéneos que cubren todas las necesidades del proyecto y de su implementación. Se espera que cada individuo tenga un perfil generalista con capacidades amplias, no siendo especialista en un ámbito exclusivo, y teniendo así la capacidad de colaborar en todo lo que esté a su alcance.
* **No se reconocen títulos específicos ni subequipos** (Schwaber y Sutherland, 2017), independientemente del área de experiencia concreta de cada miembro, de manera que todos son homogéneos desde el punto de vista de la responsabilidad que asumen en el proceso.

**El tamaño del equipo de desarrollo**

El número de desarrolladores debe ser lo bastante pequeño como para que su trabajo resulte ágil, pero lo bastante grande como para ser capaces de producir incrementos significativos en cada iteración.

* **Equipos de menos de tres personas** reducen la variedad de habilidades esperadas y la posible comunicación y flujo de información.
* **Equipos de más de 9 personas** requieren demasiado esfuerzo de coordinación y dificultan que un proceso empírico resulte de utilidad.

Además, el equipo de desarrollo tiene **tres responsabilidades fundamentales** (Alaimo, 2013):

* **Realizar estimaciones** sobre el esfuerzo de implementación de cada funcionalidad, que comparten con el *Product Owner*.
* **Comprometerse con la entrega** de las características acordadas en cada *sprint* contando con el tiempo y recursos disponibles.
* Realizar la **demostración del producto** en el momento de entrega al final de cada *sprint*.

El *Scrum Master* (SM)

El *Scrum Master* adopta una **función de *coaching*** dentro del equipo, y es el encargado de ayudar al equipo de desarrollo y al *Product Owner* a seguir la metodología de manera correcta, resolviendo cualquier impedimento que pueda surgir.

**No se debe confundir su figura con la de un jefe de proyecto**, pues no está encargado de planificar ni de asignar tareas. Lo ideal es que exista un *Scrum Master* con dedicación completa para el equipo, aunque en proyectos pequeños puede desarrollar esta labor algún miembro del propio equipo de desarrollo (Sutherland y Schwaber, 2012).

Existen diferentes **guías y listas de comprobación** que pueden ayudar al *Scrum Master* a desarrollar sus tareas de manera eficiente (CollabNet, 2012; James, 2012; Kniberg, 2010). Estos documentos pueden ayudar a este rol a realizar un correcto seguimiento de las actividades del equipo y a detectar posibles obstáculos.

2.6. Los artefactos de Scrum

El proceso de Scrum propone un número de elementos que permiten formalizar y controlar el proceso de desarrollo, de una manera ágil y ligera. **Están diseñados para maximizar la transparencia de la información clave** (Schwaber y Sutherland, 2017). Es misión de*l Scrum Master* asegurar que esta transparencia se da en la práctica, proponiendo acciones correctivas o prácticas específicas en caso de ser necesario.

En esta sección vamos a analizar estos artefactos, comprendiendo la misión y utilidad que suponen en el conjunto del proceso. **En ningún caso es recomendable que el *Scrum Master* y el *Product Owner* sean la misma persona.**

La pila del producto

**La pila del producto —*Product Backlog* (PB)— es el documento fundamental que almacena los requisitos del proyecto, como una lista ordenada y priorizada de funcionalidades que mantiene el *Product Owner****.*

Es un documento vivo, que evoluciona a medida que el desarrollo avanza y el entorno cambia. Puede haber modificaciones en requisitos existentes debidas a cambios en el negocio, el mercado o la propia tecnología, y su posible eliminación o aparición de nuevos requisitos.

Cada uno de los elementos de esta lista es un***Product Backlog Item (PBI)*** y contiene una descripción, un orden y, preferiblemente, también una estimación de complejidad y de valor respecto del producto final. Muchas veces se incluyen también descripciones de las pruebas que demostrarán su correcta implementación.

En el caso de que trabajen varios equipos en el mismo producto, se **mantiene un único PB**, pudiéndose emplear un atributo adicional que indica el equipo asignado.

**El refinamiento del *Product Backlog***

Consiste en la revisión continuada e iterativa, por parte del ***Product Owner* y del equipo de desarrollo** de los elementos del PB, en cuanto a definición, priorización y estimación de esfuerzos.

El *Product Owner* también puede introducir modificaciones de manera unilateral cuando lo considere necesario, atendiendo siempre a criterios de negocio.

En las revisiones de cada *sprint*, el *Product Owner* puede utilizar este documento para hacer un seguimiento del trabajo restante. Esta estimación es aproximada y, desde un punto de vista empírico, solo es posible utilizar la experiencia previa acumulada, puesto que en entornos complejos el futuro es a menudo incierto.

Lapila del *sprint*

**La pila del *sprint* —*Sprint Backlog* (SB)—** **está formada por el conjunto de los PBI seleccionados al comienzo de cada *sprint* para su implementación**.

Hace visible el esfuerzo requerido por parte del equipo de desarrollo para alcanzar el objetivo de la iteración y constituye el documento fundamental para su planificación. En este caso, **la responsabilidad de su mantenimiento y posible adaptación recae en el equipo de desarrollo.** Con el objetivo de facilitar la mejora continua, también se suele incluir algún elemento de mejora desde el punto de vista del proceso, identificado en la **reunión de retrospectiva** del anterior *sprint*.

Este documento permite hacer un seguimiento de la iteración comprobando la cantidad y duraciones estimadas de trabajos restantes. Este proceso de seguimiento se realiza en cada una de las reuniones diarias y permite al equipo gestionar su propio progreso.

2.7. Los eventos de Scrum

Para facilitar el control y el seguimiento del proceso, fomentar los flujos de comunicación y realizar las actividades de planificación necesarias en cada iteración, **Scrum también propone una serie de reuniones, o eventos con propósitos específicos**, distribuidas a lo largo del proceso de desarrollo. Todos estos eventos tienen una duración máxima controlada, del mismo modo que la duración de cada *sprint* es constante. **Cada *sprint* actúa como un contenedor para el resto de los eventos, y cada evento es considerado una oportunidad de mejora y adaptación**.

Los **eventos predefinidos por la metodología** son los siguientes (Schwaber y Sutherland, 2017):

El *sprint*

Según Schwaber y Sutherland (2017), **es cada uno de los períodos de tiempo de duración fija, nunca superior al mes, durante el cual se crea un incremento del producto «terminado, utilizable y potencialmente desplegable»** (p. 9).

Durante un *sprint* no se realizan cambios que puedan afectar el objetivo de ese *sprint*, pero se pueden aclarar o renegociar aspectos relativos al alcance en colaboración con el PO.

En hecho de que su duración se mantenga constante ayuda a mejorar la predictibilidad del proceso general, aprendiendo de los objetivos alcanzados y problemas surgidos en iteraciones anteriores.

Cada uno de los *sprint*s contiene **cuatro eventos fundamentales**:

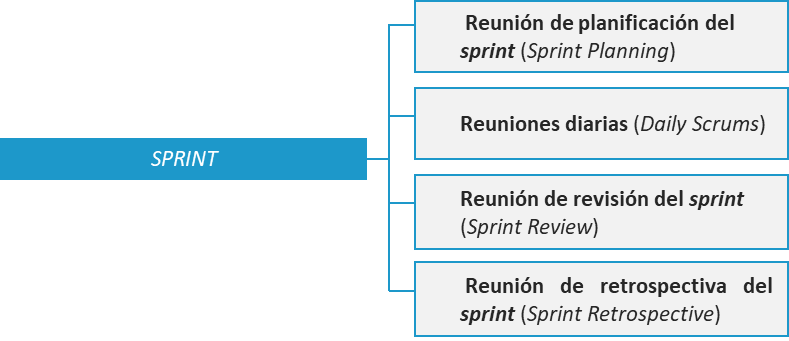


Figura 6. Eventos fundamentales

Planificación del *sprint*

**La reunión de planificación de *sprint* (*Sprint Planning*) es el punto de arranque de cada iteración.** En ella se establecen los compromisos entre el equipo de desarrollo y el *Product Owner* sobre el alcance y características de la nueva entrega.

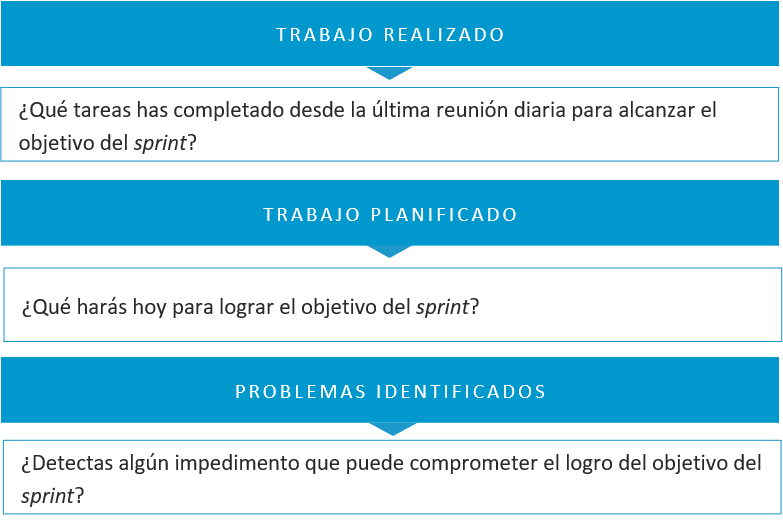
En esta reunión se debe responder a dos preguntas (Schwaber y Sutherland, 2017):

* **¿Qué puede hacerse en este *sprint*?** Es la primera fase de la reunión. Se parte del *Product Backlog,* y el *Product Owner* propone los elementos que deberían formar parte del objetivo. El número y configuración final de los PBI seleccionados depende de la capacidad disponible y la experiencia acumulada en iteraciones previas, siendo algo que debe determinar finalmente el equipo de desarrollo.
* **¿Cómo se conseguirá alcanzar el objetivo del *sprint*?** En esta segunda fase, el equipo de desarrollo analiza las funcionalidades seleccionadas y decide cómo construirá el nuevo incremento terminado. L**os PBI se descomponen en tareas individuales**, cada una de las cuales no debería tener una duración superior a un día, y los miembros del equipo se autoasignan estas tareas comprometiéndose con su desarrollo. **El resultado de este trabajo, que puede incluir tareas de diseño previo, es la pila del *sprint*** —*Sprint Backlog* (SB)—.

Scrum diario

**Los scrum diarios (*Daily Scrums*) son reuniones diarias, al comienzo de la jornada, de una duración máxima de quince minutos en la que participa el equipo de desarrollo. En ella se revisa el trabajo realizado desde la última reunión y se planifica el trabajo de la jornada.**

En estas reuniones, cada uno de los miembros del equipo de desarrollo debe responder a las siguientes preguntas:



Esta reunión es gestionada y dirigida por el propio equipo de desarrollo, y el *Scrum Master* se encarga de que se respeten los plazos y de que los flujos de comunicación sean los adecuados. Puede participar el PO para resolver dudas.

Revisión del *sprint*

**Cuando finaliza cada *sprint* se lleva a cabo la reunión de revisión del *sprint* (*Sprint Review*), que tiene por objetivo demostrar el incremento y obtener retroalimentación de los interesados en el proyecto.**

Es fundamental la participación de todos los miembros del equipo en esta reunión de carácter informal, en la que se analiza el trabajo realizado y se proponen mejoras o modificaciones sobre la pila del producto de cara al futuro.

Además del *Product Owner*, pueden participar otros *stakeholders* invitados por él. El resultado de esta reunión es la aceptación (o el rechazo) de la nueva entrega, la decisión de poner la nueva versión en producción, y una pila del producto actualizada a partir del análisis realizado.

Retrospectiva del *sprint*

Esta última reunión de cada iteración es de **carácter eminentemente técnico**, y en ella participa el equipo de desarrollo para analizar su propio trabajo y plantear posibilidades de mejora sobre los procesos.

El *Scrum Master* se encarga una vez más de controlar los tiempos y asegurarse de que el diálogo se desarrolla bajo un espíritu constructivo. Tras analizar el último *sprint,* evaluando las personas, procesos y herramientas, se crea **un plan de mejoras** que se incluirá en el siguiente *sprint*.

2.8. Las historias de usuario

Un elemento fundamental en Scrum, como en cualquier otro proceso de desarrollo de software, son los requisitos. En el caso de Scrum **los requisitos se materializan en** **historias de usuario** —*User Stories* (US)—, que son uno de los elementos integrantes del *Product Backlog* (PB) que mantiene el *Product Owner* (PO).

En un sentido amplio, el *Product Backlog* contiene PBI (*Product Backlog Items*), que son las historias de usuario y cualquier otra característica asociada al sistema que no necesariamente esté asociada a una funcionalidad o beneficio concreto para el usuario (por ejemplo, requisitos no funcionales).

La mejor manera para enunciar las historias de usuario (y en general cualquier PBI) es la siguiente:

**como <rol> quiero <funcionalidad o evento> para <beneficio>**

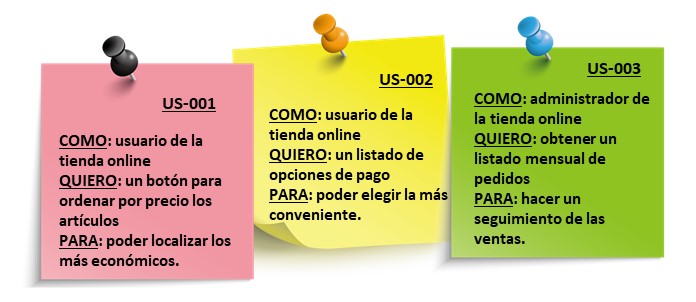


Figura 7. Algunos ejemplos de historias de usuario

En la figura 7 se muestran algunos ejemplos de historias de usuario, escritos de manera informal sobre simples pósits (figura 7). Una manera más elaborada de mantener la información del ***Product Backlog*** es mediante tablas como la que se muestra a continuación (tabla 1), donde cada historia va acompañada de una descripción detallada, los criterios de aceptación, una prioridad, y una estimación de esfuerzo.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Historia** | **Descripción** | **Criterios de aceptación** | **Prioridad** | **Esfuerzo** |
| HU1 |  |  |  |  |  |
| HU2 |  |  |  |  |  |
| HU3 |  |  |  |  |  |

Tabla 1. Plantilla para la definición de historias de usuario

Para que un PBI sea correcto es conveniente que siga la **regla INVEST** (Agile Alliance, 2015), que fue acuñada por Bill Wake, en 2003, ofreciendo **una serie de criterios que permiten evaluar la calidad de los PBI** (Wake, 2003).

* ***Independent* (independiente)**. Los PBI deberían ser independientes entre sí y no presentar solapamientos conceptuales. Esto garantiza que su definición no impone un orden de implementación, y que las estimaciones sean más sencillas.
* ***Negotiable* (negociable)**. La tarjeta que contiene cada historia ofrece una descripción general lo suficientemente detallada para que resulte comprensible. Los detalles deben ser negociados con el PO en tiempo de desarrollo (la historia captura la esencia, no los detalles).
* ***Valuable* (valiosa)**. La historia debe resultar valiosa para el PO (y en última instancia, para el cliente y usuarios).
* ***Estimable* (estimable).** Debe ser fácil para el equipo la estimación de esfuerzo asociado a cada historia. Las mayores dificultades para ello son la falta de conocimiento sobre el producto (lo cual requiere mayor negociación) o su excesivo tamaño (y en este caso debemos descomponer la historia).
* ***Small* (pequeña)**. Idealmente cada historia debería ser implementada por dos o tres personas trabajando una semana.
* ***Testable*****(testable o verificable)**. Como cualquier requisito, cada historia debería estar asociada a algún tipo de prueba que permita verificar que ha sido correctamente implementada. Reflexionar sobre las pruebas puede ayudar a comprender el verdadero alcance del requisito o, incluso, a identificar historias que no son necesarias. Los requisitos no funcionales también deberían ser verificables.

Como sabemos, en cada iteración los PBI se descomponen en tareas concretas, más manejables, y que pueden ser abordadas por desarrolladores específicos en función de su experiencia. Wake también ofrece algunas indicaciones sobre las características deseables de cada una de estas tareas, que deben ser en este caso **SMART**:

* ***Specific*** **(específica)**. Debe ser enunciada en términos muy concretos y comprensibles, y garantizar que no tiene solapamientos con otras tareas.
* ***Measurable* (medible)**. Debe ir asociada con algún tipo de métrica que permita garantizar que la tarea está completada.
* ***Achievable* (alcanzable)**. Las tareas son asignadas a miembros concretos del equipo (o a parejas, en el caso de la programación en parejas). Completarlas debe suponer un objetivo realista teniendo en cuenta los recursos disponibles y la formación y experiencia del equipo.
* ***Relevant* (relevante)**. Aunque la descomposición en tareas es importante desde el punto de vista técnico, cada una de las tareas debe ser justificable a ojos del PO, representando algún tipo de contribución necesaria y significativa para el PBI.
* ***Time-boxed* (acotada en el tiempo)**. La duración estimada de la tarea es un indicador de su complejidad, y permite al desarrollador asignado solicitar ayuda en caso de que observe que no cumplirá los plazos.

2.9. Extensiones de Scrum

Scrum ofrece un **marco de trabajo general**, define roles, artefactos y actividades que guían el proceso de desarrollo a través de iteraciones de duración constante.

Este marco es apropiado para **combinarse con elementos de otras metodologías**. Veamos dos ejemplos.

Scrumban: combinación de Scrum y Kanban

El **método Kanban** hace énfasis en la entrega continua de resultados, teniendo en cuenta que se deben evitar saturaciones y cuellos de botella según las capacidades del equipo de desarrollo.

Se basa en **tres principios** (McLaughlin, 2018):

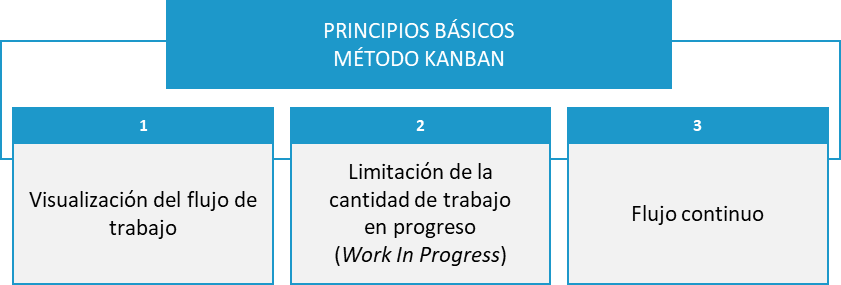


Figura 8. Plantilla para la definición de historias de usuario

* **Visualización del flujo de trabajo**. Ofrece una herramienta fundamental, **el tablero Kanban,** que permite obtener información visual sobre cada tarea y sus relaciones con un solo golpe de vista.

Se descompone el trabajo en tareas que se desplazan de izquierda a derecha, y en columnas se muestra el estado del trabajo (pendiente, en progreso, finalizado, en la configuración más básica).

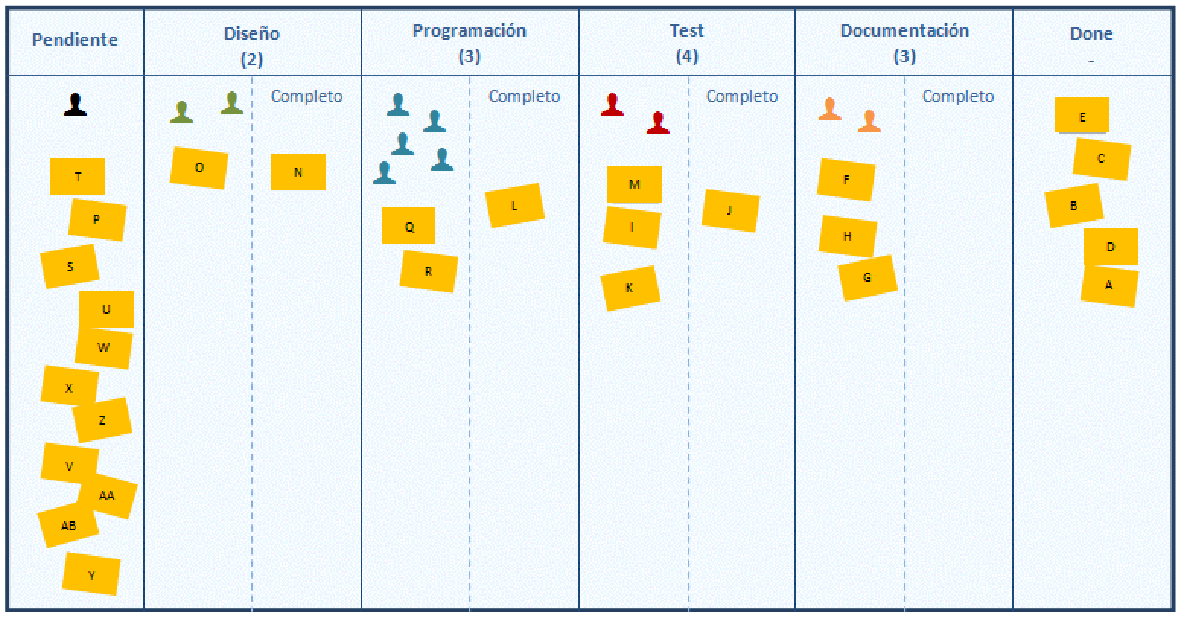


Figura 9. El tablero Kanban. Fuente: Vila (2016)

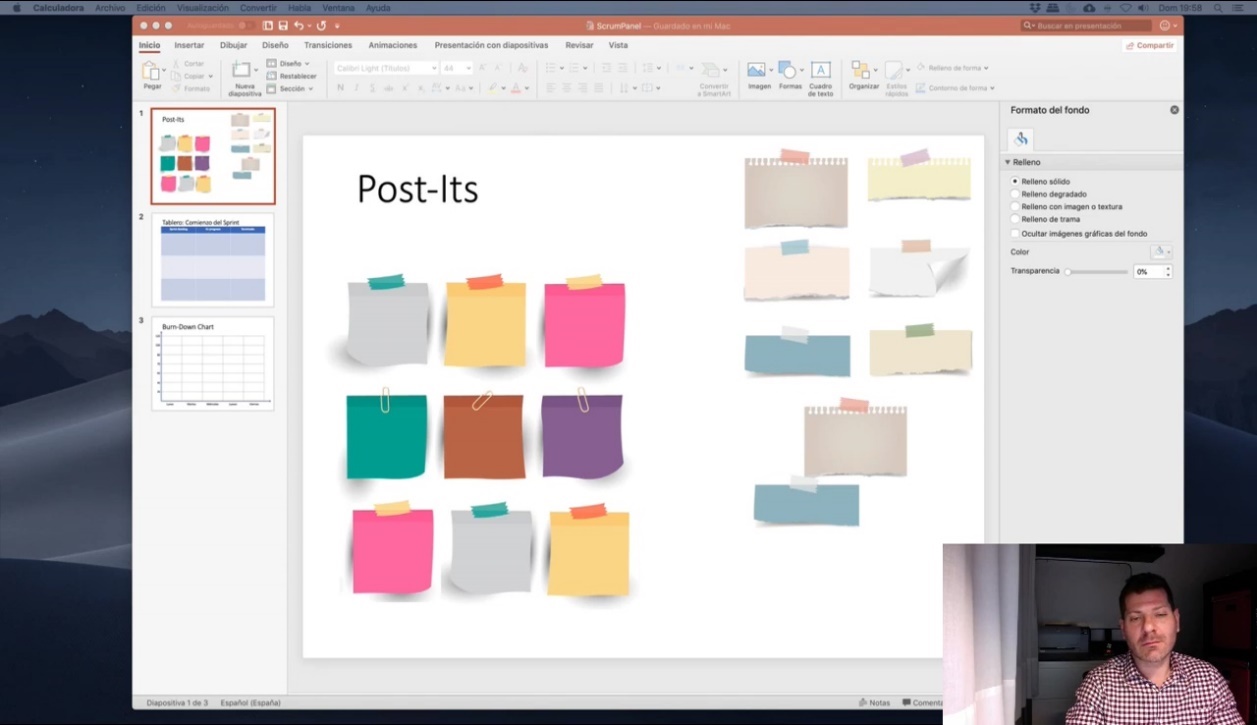
* **Limitación de la cantidad de trabajo en progreso —*Work In Progress* (WIP)—**. Busca equilibrar la carga de trabajo en cada momento, de manera que el equipo no quede saturado. En cada columna del tablero no se puede superar un volumen de trabajo prefijado.
* **Flujo continuo**. Cuando una tarea se finaliza, la siguiente más prioritaria se pone de inmediato en desarrollo. No se trabaja por lotes de trabajo, como sucede en Scrum con sus *sprint*s, sino que todas las tareas pendientes son susceptibles de ser comenzadas en cualquier momento del desarrollo.

La manera más simple de combinar ambas metodologías **consiste en aprovechar las capacidades de visualización del proceso que ofrece Kanban**, con el tablero Kanban, como podemos ver en la figura 9.

En Scrumban realmente se tiende a adoptar un modelo de proceso con flujo continuo, como propone Kanban, de manera que la pila del *sprint* ya no es necesaria y todas las tareas pendientes están agrupadas en la columna inicial del tablero (Vila, 2016).

**Vídeo: El tablero de Scrumban**

En este vídeo vamos a hacer una pequeña introducción a lo que sería trabajar con la metodología ágil Scrum, es decir, trabajar a lo largo del desarrollo de un proyecto a partir de un tablero que nos servirá para representar las historias de usuario y las tareas implementadas en cada iteración.



Accede al vídeo a través del aula virtual.

Estos vídeos estarán disponibles en la plataforma para colombia????

Combinación de Scrum y XP

Como hemos visto, Scrum no prescribe la utilización de prácticas de ingeniería concretas y se reduce a proporcionar un marco de trabajo general para gestionar el proceso. Sin embargo, es habitual que muchos equipos adopten progresivamente prácticas concretas, como las **12 prácticas propuestas por la Programación Extrema** (Beck, 2000). Entre ellas podemos citar la programación en parejas, el empleo de estándares de programación o la propiedad del código compartida.

2.10. Estimaciones ágiles

Uno de los **factores críticos en la planificación de cada *sprint*** es la **definición de su objetivo**, dependiente de la carga de trabajo que el equipo puede asumir.

En proyectos de desarrollo de software la incertidumbre inicial es muy grande, y se va reduciendo a medida que avanza el proyecto siguiendo una secuencia denominada **«cono de incertidumbre de las estimaciones»,** como se observa en la figura 10.

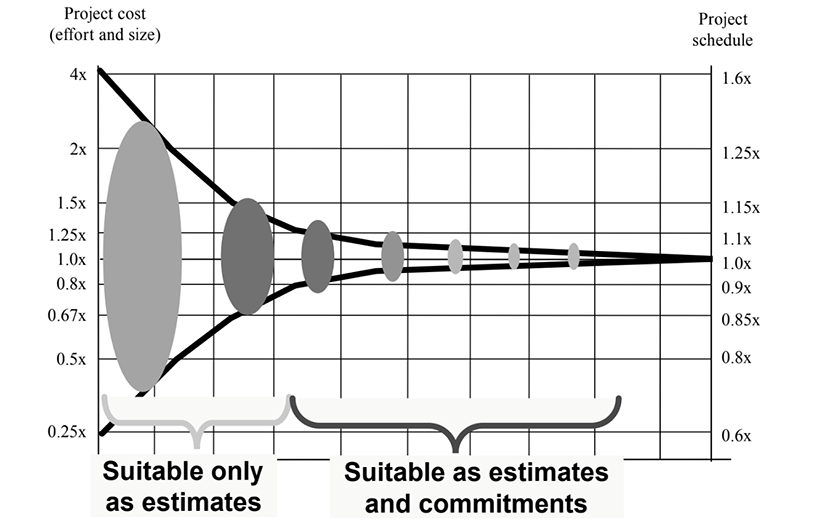


Figura 10. El cono de incertidumbre de las estimaciones de proyectos de software.  
Fuente: McConnell (2008)

**Es necesario contar con herramientas que permitan estimar de algún modo la complejidad de implementación de los requisitos y facilitar su seguimiento.**

Estimaciones de los PBI y las tareas

Es evidente que **realizar estimaciones precisas en horas de trabajo es complejo**, más aún en proyectos complejos con gran incertidumbre. Para ello, es habitual seguir la **siguiente técnica**:

* **Estimar la duración de las historias de usuario en puntos de historia** (*Story Points*). Para ello, el equipo de desarrollo dialoga con el PO para comprender a fondo cada historia y, entre todos, asignan a cada una de ellas una puntuación tomando valores del conjunto {0, 1, 3, 5, 8, 13, 21, 40, 100} (Alaimo, 2013). Esto da una idea al menos de la complejidad relativa de las historias del PB.
* **Estimar las tareas en horas de trabajo**. Durante la reunión de planificación del *sprint*, se realiza una descomposición de las historias del *Sprint Backlog* en tareas concretas de implementación. En este momento es más fácil realizar estimaciones más precisas para cada una de las tareas o actividades individuales y, así, establecer la carga de trabajo concreta para cada *sprint* individual.

Para realizar estas estimaciones se pueden emplear técnicas como el «**Método Delphi»** o el **«Póker de Planificación»** (Alaimo, 2013; Albaladejo, 2009). La información obtenida complementa las tarjetas de historia de usuario, como podemos ver en la figura 11.



Figura 11. Tarjeta de historia con estimación de esfuerzo, prioridad y nivel de riesgo.   
Fuente: Albaladejo (2009)

Seguimiento del *sprint*

La suma de horas de trabajo asignadas a todas las tareas que forman parte de un *sprint* permite determinar la **carga total de trabajo**, teniendo en cuenta las capacidades del equipo de desarrollo (número de integrantes, horas de dedicación, etc.).

El seguimiento del progreso de desarrollo se realiza habitualmente empleando un artefacto adicional: el ***Burndown Chart*** (Kniberg, 2015). Este gráfico muestra en abscisas (eje horizontal) los días que componen el *sprint*, y en ordenadas (eje vertical) las horas de trabajo correspondientes a las tareas que restan por implementar.

El primer día del *sprint*, antes de la reunión diaria, el *Burndown Chart* puede tener la apariencia mostrada en la figura 12, con todas las tareas aún sin comenzar y el gráfico representando únicamente una estimación de progreso lineal.

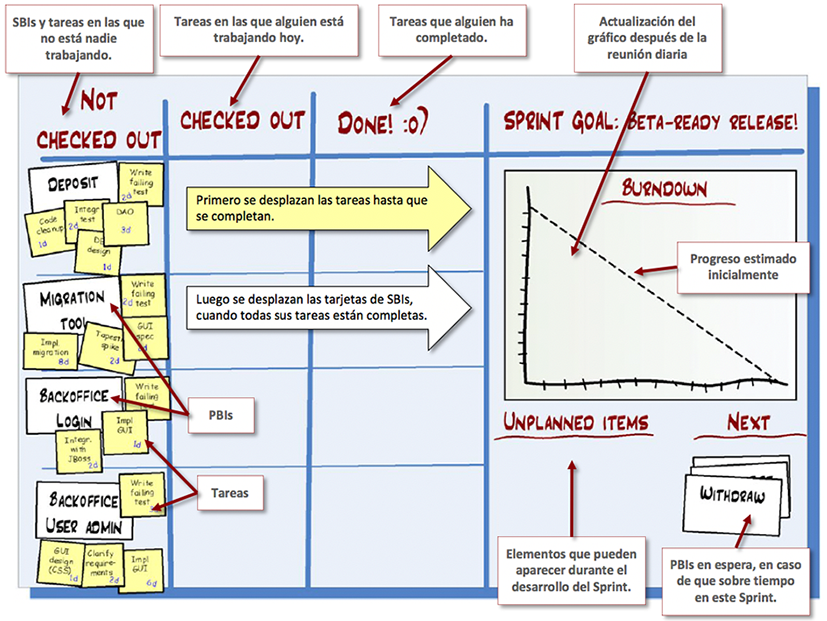


Figura 12. Configuración del panel y el *Burndown Chart* al comienzo de un *sprint*.  
Fuente: Elaboración propia a partir de Kniberg (2015)

Al final de cada Scrum diario, el equipo modifica el panel con las tareas que finalizó el día anterior, y es el momento de actualizar el gráfico de progreso con la suma de horas correspondientes a todas estas tareas.

Una posible situación en mitad de un *sprint* se representa en la figura 13. Como podemos ver, se ha completado una historia junto con todas sus tareas. Otras tres historias están en progreso, y en el momento actual solo hay en desarrollo 3 tareas. Además, han aparecido 3 tareas imprevistas, y el gráfico de avance sigue aproximadamente la evolución planificada.

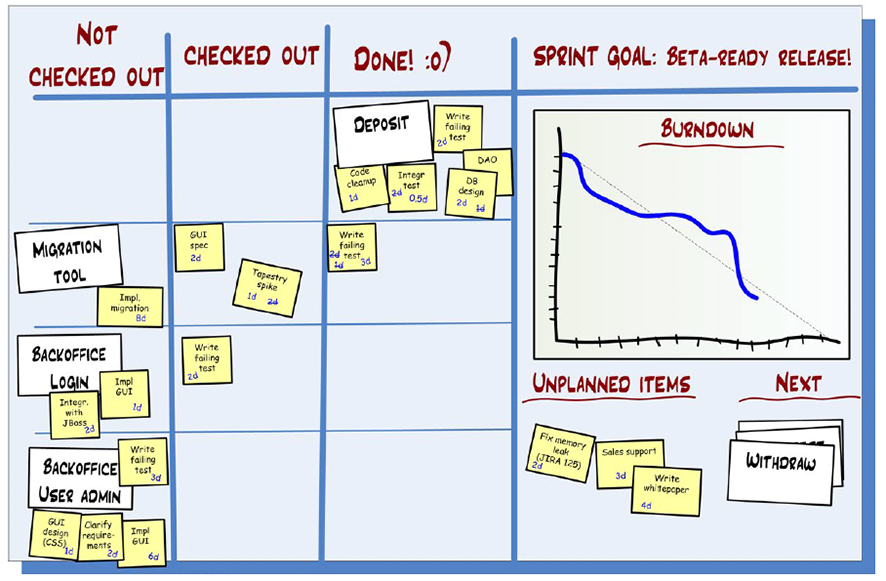


Figura 13. Configuración del panel y el *Burndown Chart* en la mitad de un *sprint*.  
Fuente: Kniberg (2015)

2.11. Escalado de Scrum

Scrum es una metodología pensada para equipos de desarrollo pequeños, donde la transparencia de la información y la comunicación habitual y fluida son posibles.

Existen **proyectos de grandes dimensiones** donde es necesaria la participación de equipos más voluminosos. En estos casos, el enfoque general es la **distribución del personal en varios equipos de Scrum**, definiendo **cauces de comunicación adicionales** que favorecen la aplicación del método general. Henrik Kniberg (2015) ha participado en proyectos con equipos de hasta 40 personas y ofrece una serie de indicaciones para gestionar este problema.

La distribución de los equipos

**Trabajar con equipos de gran tamaño introduce dificultades**, como la excesiva duración de las reuniones diarias, o la falta de visión general de los miembros del equipo por falta de comunicación. Cuando trabajamos con **equipos experimentados** y es factible dividir el desarrollo en varias líneas diferenciadas que no comparten código, es posible la **división del equipo**. En otro caso suele ser preferible mantener un único equipo.

Dada una conformación de equipos, el *Scrum Master* debe observar cómo se establecen los **cauces de comunicación entre sus miembros.** En la figura 14 observamos dos posibles situaciones:

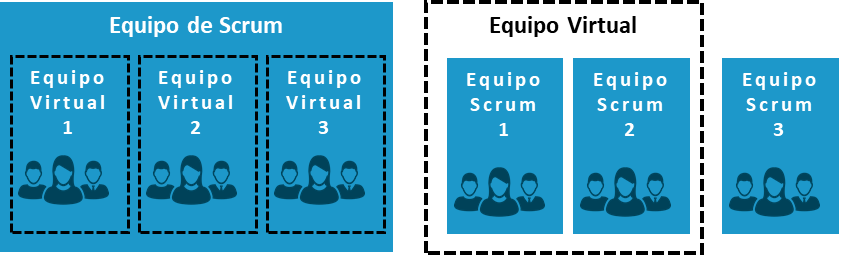


Figura 14. Formación de emergente de equipos virtuales.  
Fuente: Elaboración propia a partir de Kniberg (2015)

* **A la izquierda** podemos ver un proyecto con un único equipo grande, pero los participantes tienden a agruparse en sus conversaciones en dos o más equipos virtuales de tamaño reducido.
* **A la derecha**, por el contrario, se ha definido un total de 3 equipos de trabajo y, sin embargo, dos de ellos tienden a establecer entre sí interacciones más frecuentes.

Si estos comportamientos emergentes de agrupación en equipos virtuales se prolongan en el tiempo puede ser un síntoma de que la agrupación no es la correcta. Se debe comentar en la reunión de **retrospectiva del *sprint*** y dejar que los equipos decidan por sí mismos. En todo caso, es recomendable no superar las 8 o 9 personas por equipo.

Otro factor es el **grado de especialización de cada uno de los equipos**.

Podríamos pensar que es preferible **agrupar individuos con conocimientos tecnológicos similares**, especializados en el desarrollo de componentes o subsistemas concretos. Esta situación aparece representada en la mitad izquierda de la figura 15, pero, como vemos, puede suceder que haya historias de usuario que afecten a varios componentes, lo que obliga a una colaboración más estrecha entre diferentes equipos, que puede ser difícil de gestionar e introduce dependencias innecesarias.

A la derecha de la misma figura observamos un enfoque con **equipos multidisciplinares**, donde cada uno por sí mismo es capaz de implementar en su totalidad cualquiera de las historias de usuario planteadas. La elección dependerá en cada caso del tamaño de los diferentes componentes, sus interdependencias, y la variedad de perfiles y especialidades con que contemos en el equipo.

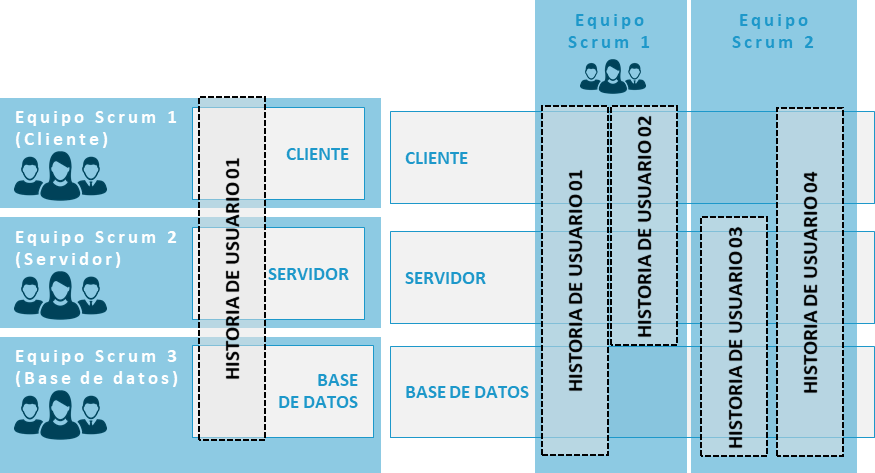


Figura 15. Especialización o multifuncionalidad de los equipos de desarrollo.  
Fuente: Elaboración propia a partir de Kniberg (2015)

La sincronización de los *sprint*s

Cuando tengamos **varios equipos trabajando en paralelo en el mismo proyecto**, es necesario determinar una manera de **sincronizar sus flujos de trabajo**.

La recomendación es que trabajemos con *sprint*s sincronizados, con todos los equipos comenzando y finalizando sus iteraciones al mismo tiempo.

Las ventajas de este enfoque son las siguientes:

1. **Disponemos de un punto natural en el que realizar redistribuciones de los equipos**, sin que ello entorpezca el normal funcionamiento de uno de ellos.
2. **Se fomenta la colaboración entre equipos**, que pueden comunicarse entre sí a la hora de planificar los objetivos de cada *sprint*.
3. **Se reduce el número de reuniones y la carga administrativa**. Las reuniones de revisión se pueden tener de manera conjunta, realizando una única demostración.

Scrum de Scrums

Cuando trabajamos en **proyectos grandes con varios equipos Scrum trabajando simultáneamente**, es necesario establecer cauces de comunicación y sincronización entre los diferentes equipos.

El Scrum de Scrums es una reunión periódica, frecuentemente diaria y con posterioridad al Scrum diario de cada equipo, en la que uno o dos representantes de cada uno de ellos acude para compartir información.

Es preferible que se trate de representantes con perfil técnico, con capacidad para decidir sobre el proceso de desarrollo y asumir compromisos, y con una visión general de la iteración actual. En algunos casos el enviado es precisamente el *Scrum Master* de cada equipo, aunque no sea lo idóneo.

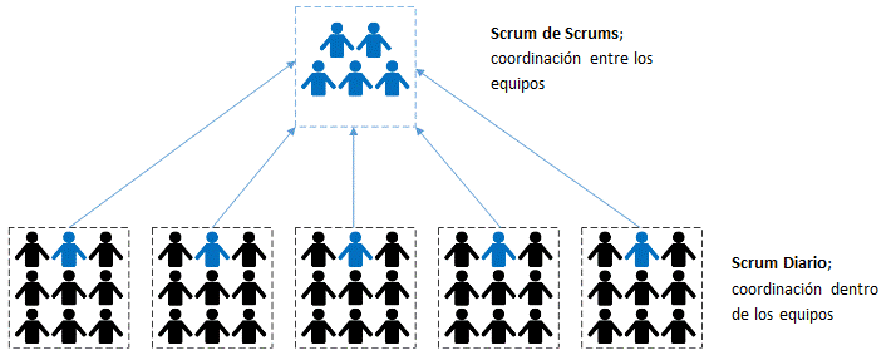


Figura 16. Representación jerárquica del Scrum de Scrums.  
Fuente: Ferrer (2016)

En estas reuniones, los representantes de los equipos deben responder a las siguientes preguntas (tres de ellas similares a las planteadas en el *Daily Scrum*):

1. ¿Qué tareas ha realizado tu equipo desde la última reunión?
2. ¿Qué tareas va a realizar tu equipo hasta la siguiente reunión para lograr el objetivo del *sprint*?
3. ¿Existe algún impedimento que afecte de manera negativa a tu equipo?
4. ¿Existe algún factor con origen en tu equipo que puede afectar a los objetivos del resto de equipos?

La gestión del *Product Backlog*

Cuando tenemos varios equipos de Scrum, cada uno de ellos con su *Scrum Master*, existen tres estrategias a la hora de gestionar los requisitos del sistema en su totalidad:

1. **Un PO y un PB.** Este es el enfoque preferido. Toda la información sobre requisitos del producto está centralizada, y el *Product Owner* tiene una visión global del proceso de desarrollo. Las reuniones de planificación de cada *sprint* se suelen realizar en el mismo lugar y momento, pero en espacios diferenciados para los diferentes equipos. Es un trabajo intenso, pero al mismo tiempo se fomenta la comunicación entre todos los participantes.
2. **Un PO y varios PB**. El mayor problema de este enfoque consiste en la asignación de historias a los diferentes equipos. Puesto que el PB es gestionado por el PO, se espera que sea este rol quien haga la asignación, pero esta es una tarea que requiere en general de un perfil técnico.
3. **Varios PO, cada uno con su PB.** Este enfoque es el más problemático, pues se diluye la visión general del producto entre varios PO que deben coordinarse entre sí. Además, en el caso de que los diferentes PB tengan bases de código comunes, se puede dificultar el proceso de desarrollo.

En un **caso ideal** en el que los diferentes equipos estuvieran asignados a componentes con distintas bases de código, estaríamos próximos a un escenario en el que el producto se descompone en varios subproyectos relativamente independientes, y solo en este caso este enfoque sería adecuado.

2.11. Referencias bibliográficas

Agile Alliance. (2015, diciembre 17). *What does INVEST Stand For?* [Blog corporativo]. Recuperado de <https://www.agilealliance.org/glossary/invest/>

Alaimo, D. M. (2013). *Proyectos ágiles con Scrum*. Buenos Aires: Ediciones Kleer.

Albaladejo, X. (2009). *Introducción a la estimación y planificación ágil* [Web]. Recuperado de <https://proyectosagiles.org/2009/06/08/introduccion-estimacion-planificacion-agil/>

Beck, K. (2000). *Extreme Programming Explained. Embrace Change*. Boston: Addison-Wesley.

CollabNet. (2012). ScrumMaster Checklist [Web]. Recuperado de <https://www.collab.net/resources/scrummaster-checklist>

Ferrer, M. (2016). Escalando equipos Scrum. Scrum de Scrums de Scrums [Web]. Recuperado de <http://managementplaza.es/blog/escalando-equipos-scrum/>

James, M. (2012). The ScrumMaster Checklist[Web]. Recuperado de <http://scrummasterchecklist.org/>

Kniberg, H. (2010). Scrum Checklist [Web]. Recuperado de <https://www.crisp.se/gratis-material-och-guider/scrum-checklist>

Kniberg, H. (2015). *Scrum and XP from the trenches: how we do Scrum* (2ª ed.). C4Media.

McConnell, S. (2008). 10 Deadly Sins of Software Estimation [presentación]. Recuperado de <http://www.austin-cs.org/archive/10-Deadly-Sins-of-Software-Estimation>

McLaughlin, M. (2018). What is Agile Methodology? [Web]. Recuperado de <https://www.versionone.com/agile-101/agile-methodologies/>

Schwaber, K. y Sutherland, J. (2017). La Guía Definitiva de Scrum: Las Reglas del Juego [Web].

Recuperado de [https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-Spanish-European.pdf#zoom=100](#zoom=100)

Sutherland, J. y Schwaber, K. (2012). *The Scrum Papers: Nuts, Bolts, and Origins of an Agile Process* (Version 1.1). Massachusetts: Scrum Inc.

Vila, J. L. (2016). Scrumban [Web]. Recuperado de <http://managementplaza.es/blog/scrumban/>

Wake, B. (2003).INVEST in Good Stories, and SMART Tasks[Blog personal]. Recuperado de <https://xp123.com/articles/invest-in-good-stories-and-smart-tasks/>

A fondo

**Lección magistral: Scrum: roles, reuniones y modelo de proceso**

En esta lección magistral se presentan las principales características de la metodología ágil de desarrollo de software Scrum. En primer lugar, comentaremos los principales roles que intervienen en el proceso. A continuación, explicaremos cuáles son las principales reuniones que se proponen y que sirven para garantizar los correctos flujos de información dentro de un proyecto. Finalmente, revisaremos el modelo de proceso en su conjunto, de manera que quede claro cómo se organiza el trabajo y cómo los anteriores elementos, roles y reuniones, encajan en el desarrollo del proyecto.



Accede a la lección magistral a través del aula virtual.

**La Guía Definitiva de Scrum: Las Reglas del Juego**

Los creadores de Scrum mantienen una guía actualizada periódicamente sobre la metodología Scrum. En ella se describe el equipo, los eventos, los artefactos y los principales valores del método.

Accede al documento a través del aula virtual o desde la siguiente dirección web: [https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-Spanish-European.pdf#zoom=100](#zoom=100)

**Scrum y XP desde las trincheras**

Kniberg, H. (2007). *Scrum y XP desde las trincheras. Cómo hacemos Scrum*. C4Media.

Este manual contiene una serie de indicaciones y mejores prácticas para enfocar el proceso de desarrollo bajo una metodología ágil como Scrum. Se ofrecen orientaciones para planificar los *sprints*, elaborar las pilas de producto y de *sprint*, enfocar las diferentes reuniones o escalar la metodología cuando se trabaja con varios equipos. También se ofrece una propuesta de combinación de Scrum con XP.

Accede al documento a través del aula virtual o desde la siguiente dirección web:

<https://goo.gl/A1yyHo>

**Scrum Inc.: Scrum Papers**

Scrum Inc. Es una empresa fundada por Jeff Sutherland, uno de los creadores de Scrum, que ofrece formación y consultoría para otras organizaciones e individuos sobre la aplicación de esta metodología. En su página web mantienen además recursos abiertos y un repositorio de artículos publicados por Sutherland, con casos de éxito e información sobre la evolución y futuro de Scrum.



Accede a la página web a través del aula virtual o desde la siguiente dirección web:

<https://www.scruminc.com/scrum-papers/>

**Scrum en 10 minutos**

En este vídeo se presenta una visión general de Scrum, explicando las historias de usuario, los principales roles, reuniones y artefactos, incluyendo el *Burndown Chart*.



Accede al vídeo a través del aula virtual o desde la siguiente dirección web:

https://youtu.be/PlLHc60egiQ

**Metodologías ágiles. El proceso SCRUM**

Seminario de formación sobre metodologías ágiles en desarrollo de software de banca, impartido por el Profesor Juan Carlos Yelmo, de la Universidad Politécnica de Madrid. En el vídeo se realiza una revisión de los procesos ágiles y una descripción detallada de Scrum, incluyendo roles, reuniones y artefactos.



Accede al vídeo a través del aula virtual o desde la siguiente dirección web:

<https://youtu.be/p9MYRrQEOGI>

**Plantillas Scrum: Historias de usuario y criterios de aceptación**

Plantillas de historias de usuario y criterios de aceptación para descarga desde pmoinformatica.



Accede al recurso a través del aula virtual o desde la siguiente dirección web:

<http://www.pmoinformatica.com/2012/10/plantillas-scrum-historias-de-usuario.html>

Actividades

Laboratorio. Especificación de requisitos de un proyecto

En esta actividad vamos a realizar la especificación de requisitos de un proyecto realizado bajo la metodología Scrum. La aplicación consistirá en una aplicación cliente para la red social Twitter.

**La actividad abarcará la especificación de requisitos del sistema completo, la elaboración de una pila historias para el primer *sprint*, y la simulación de la iteración durante los tres primeros días del desarrollo.**

**Objetivos de la actividad**

Realizando esta actividad alcanzarás los siguientes objetivos:

* Saber enunciar historias de usuario correctas que representen las funcionalidades esperadas del producto.
* Comprender la importancia y naturaleza de la pila de producto como elemento fundamental en el desarrollo bajo Scrum.
* Saber descomponer las historias del *Sprint Backlog* en tareas concretas para su desarrollo.
* Comprender el funcionamiento de la metodología Scrum simulando el desarrollo del primer *sprint* del proyecto.
* Aprender a utilizar el *Product Backlog*, el *Sprint Backlog*, el tablero Kanban y el *Burndown Chart*.
* Valorar la importancia de la comunicación oral entre los implicados en el desarrollo a la hora de definir los requisitos y el alcance del proyecto.

**Descripción de la actividad**

**Durante el laboratorio, el profesor adoptará el rol de *Product Owner*. Los alumnos actuarán como posibles clientes y usuarios de la aplicación, y de manera colaborativa, se desarrollará una especificación completa del sistema en forma de historias de usuario.**

El profesor, como PO, preguntará a los alumnos acerca de las posibles funcionalidades que sería conveniente incluir en la aplicación. No se crearán grupos para esta actividad, y todos los asistentes participarán en la conversación asumiendo el rol de posibles usuarios de la aplicación. En un documento colaborativo, el profesor irá escribiendo las historias de usuario en un formato adecuado, dando lugar a la especificación del sistema. Esta especificación quedará disponible para que los alumnos puedan completar la actividad con ella, utilizándola como base.

Este conjunto de historias de usuario constituirá el *Product Backlog* (PB), que será utilizado por todos los alumnos para desarrollar el resto de la actividad. En el PB las historias quedarán priorizadas desde el punto de vista del negocio, y su complejidad quedará estimada mediante puntos de historia.

Con el PB común, cada alumno deberá definir el *Spring Backlog* seleccionando un subconjunto de historias. Cada historia deberá descomponerse en tareas de desarrollo concretas, asignando una estimación de duración de implementación en horas a cada una de ellas. Con el *Sprint Backlog* definido se simularán los tres primeros días de la iteración, representando el panel y el *Burndown Chart* correspondientes tal como quedarían al finalizar la reunión diaria.

**Entrega**

Cada alumno entregará de manera individual un documento en formato PDF que deberá contener los siguientes apartados:

1. **Introducción**. Describir el proyecto, sus objetivos y características. También se debe describir la composición del equipo de desarrollo (número de desarrolladores y dedicación en horas/día), puesto que esta información será relevante para planificar el *sprint*. Extensión: media página.
2. ***Product Backlog***. En este apartado se mostrará la pila de producto construida durante el laboratorio. Para los alumnos que no puedan asistir presencialmente al laboratorio, esta información quedará colgada en el foro de la asignatura.
3. ***Sprint Backlog***. En esta sección se indicarán los PBI seleccionados para su implementación en el primer *sprint*, y la descomposición de cada uno de ellos en tareas individuales.
4. **Primer panel y *Burndown Chart*.** Representación del tablero y el gráfico al finalizar la primera reunión diaria del *sprint*. Se deberá explicar brevemente la construcción de ambos elementos.
5. **Segundo panel y *Burndown Chart***. Representación del tablero y el gráfico al finalizar la segunda reunión diaria del *sprint*. Se deberá explicar brevemente la construcción de ambos elementos, en función de la evolución representada.
6. **Tercer panel y *Burndown Char****t*. Representación del tablero y el gráfico al finalizar la tercera reunión diaria del *sprint*. Se deberá explicar brevemente la configuración de ambos elementos, en función de la evolución del proyecto.

Para la representación de los paneles se recomienda utilizar una plantilla similar a la que se muestra a continuación, donde se han utilizado pósit grandes para los PBI, y otros más pequeños para indicar las tareas. También se pueden emplear formas de Microsoft Word.

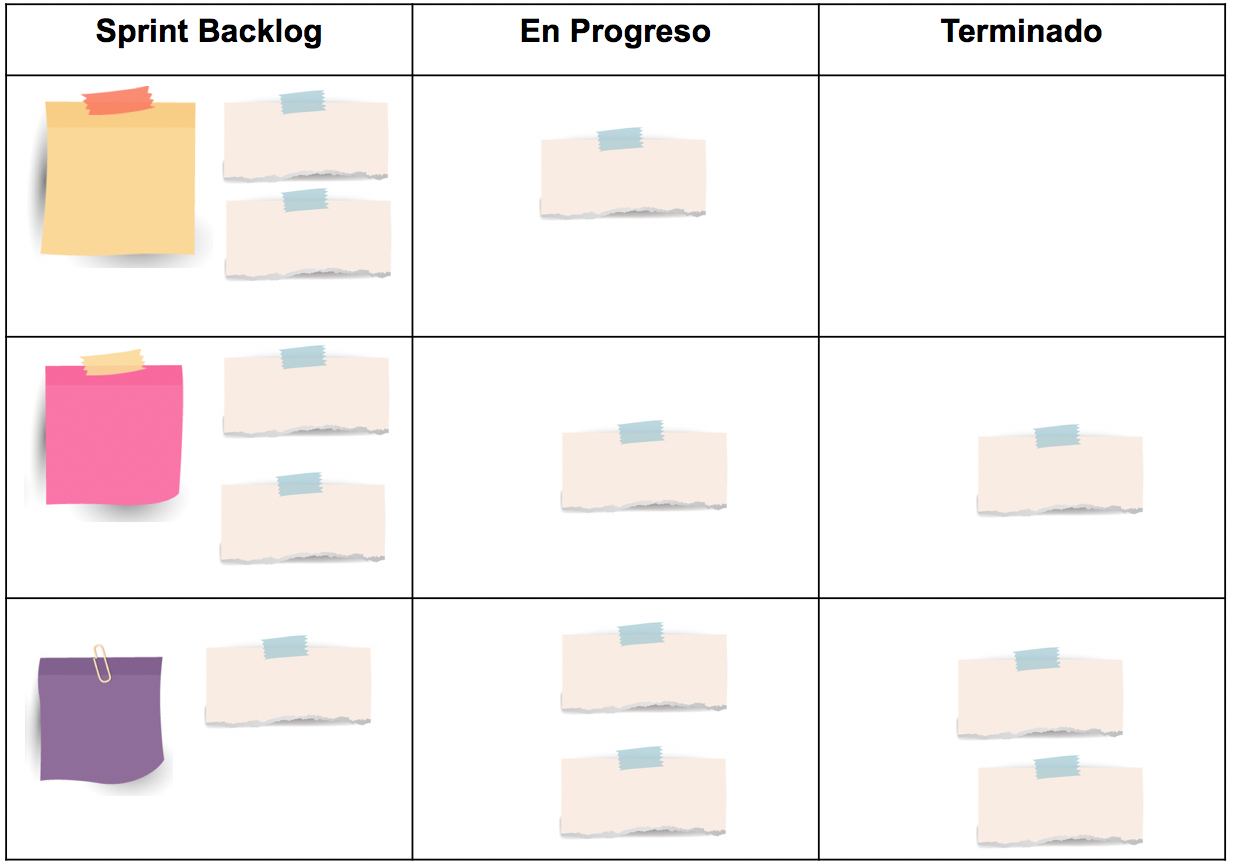


Tabla 1. Modelo de plantilla para la representación de los paneles.   
Fuente de las imágenes de pósit para PBI:   
<https://pngtree.com/freepng/vector-color-sticky-notes_1443970.html>  
Fuente de las imágenes de pósit para tareas:   
<https://pngtree.com/freepng/paper-tearing-effect-sticker_3326453.html>

**Criterios de evaluación**

Al evaluar esta actividad se tendrán en cuenta los siguientes aspectos:

* Correcta estructura y redacción del documento, sin faltas ortográficas y con capacidad de síntesis.
* Estética del documento, cuidando los aspectos visuales de los paneles y gráficos representados.
* Correcta definición de la pila del *sprint* y descomposición en tareas.
* Correcta utilización del panel Kanban.
* Correcta construcción de los *Burndown Charts*, representando el avance de la iteración.

Test

1. ¿Qué elementos fundamentales define Scrum para guiar el proceso de desarrollo?

A. Roles, reuniones y artefactos.

B. Reuniones, prácticas y roles.

C. Procesos, reuniones y artefactos.

D. El *Burndown Chart* y la pila de producto.

1. ¿Cuáles son las responsabilidades del *Product Owner* en Scrum?

A. Comunicarse con el cliente y otros *stakeholders* para obtener información sobre el producto y sus intereses.

B. Definir los requisitos del producto que mantiene en la pila de producto.

C. Determinar el alcance de cada iteración teniendo en cuenta las prioridades de negocio y la opinión del equipo de desarrollo.

D. Todas las respuestas anteriores son correctas.

1. ¿Cuál de las siguientes es una característica deseable de un equipo de desarrollo en Scrum?

A. Cada miembro del equipo debe estar muy especializado en un área de desarrollo concreta.

B. Son equipos autorganizados y multifuncionales, con perfiles heterogéneos que planifican su propio trabajo sin que exista la figura del *Project Manager.*

C. Se comunican directamente con el cliente para comprender los requisitos.

D. Son siempre de gran tamaño, por encima de las 20 personas.

1. ¿Qué artefacto resulta de la reunión de planificación de un *sprint*?

A. La pila del producto.

B. El *Burndown Chart*.

C. La pila del *sprint*.

D. Ninguna de las respuestas anteriores es correcta.

1. ¿Quiénes son los participantes del Scrum Diario?

A. El equipo de desarrollo y el *Scrum Master* y, opcionalmente, el *Product Owner* que puede resolver dudas.

B. Únicamente el equipo de desarrollo.

C. El *Scrum Master* y el *Product Owner*, que revisan el trabajo y ajustan la planificación.

D. El *Product Owner* y el equipo de trabajo.

1. ¿Cuál es el propósito de la reunión de revisión del *sprint*?

A. Demostrar el incremento ante los *stakeholders*, para que el *Product Owner* lo acepte o lo rechace.

B. Obtener retroalimentación de los interesados en el proyecto, que pueden comprobar el resultado del incremento y valorarlo.

C. Reflexionar sobre la práctica realizada y el desarrollo de la iteración.

D. Las respuestas A y B son correctas.

1. ¿Cuáles de las siguientes características debe poseer una buena historia de usuario?

A. INVEST: innovadora, negociable, verificable, entendible, sencilla y temporalizable.

B. INVEST: independiente, negociable, valiosa, estimable, pequeña y verificable.

C. SMART: específica, medible, alcanzable, relevante y acotada en el tiempo.

D. SMART: sencilla, manejable, asequible, razonable y temporalizable.

1. ¿Qué herramienta visual procedente de Kanban se utiliza habitualmente en proyectos Scrum?

A. Las tarjetas de historias de usuario.

B. El *Burndown Chart*.

C. El tablero Kanban.

D. Ninguna de las respuestas anteriores es correcta.

1. ¿Cómo podemos estimar la complejidad de implementación de las historias de usuario en Scrum?

A. Empleando puntos de historia y técnicas como el «póker de planificación».

B. Las historias de usuario siempre deben estimarse de manera precisa en horas de trabajo.

C. Preguntando al *Product Owner* sobre la complejidad de implementación de cada historia.

D. No es necesario estimar la duración de implementación de cada historia. Si no se puede completar en un *sprint* se continúa en el siguiente.

1. ¿Quién debería detectar la emergencia de equipos virtuales en un proceso Scrum?

A. El jefe de proyecto.

B. El *Product Owner*.

C. El *Scrum Master*.

D. Ninguno de los anteriores.