Módulo 1: Metodologías Ágiles - Scrum

AUTORES: DETKE, Ramiro Fernando, FINOCHIETTO, Jorge Manuel

Índice

Introducción

Objetivos:

- Introducir el desarrollo de software como un proceso que requiere de una metodología de trabajo en equipo que garantice un producto acorde a las necesidades del usuario que lo demanda
- Comprender los conceptos de la metodología Scrum, sus componentes y su implementación a través de ciclos de corta duración y de una planificación consensuada por todo el equipo de trabajo,
- Analizar y definir las necesidades del usuario a través de la especificación de requerimientos utilizando historias de usuario

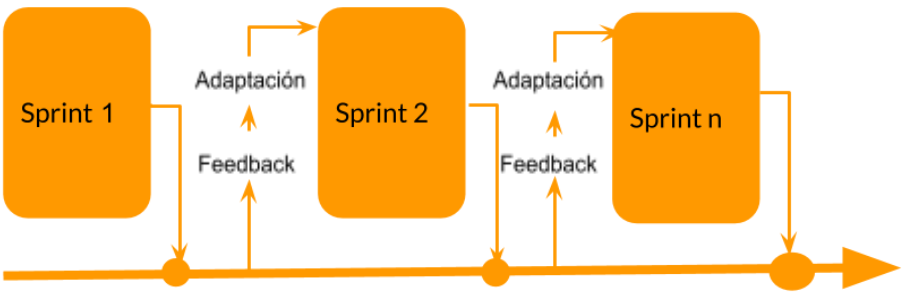
Desarrollo de Software

Según la IEEE, el software es el "conjunto de los programas de cómputo, procedimientos, reglas, documentación y datos asociados, que forman parte de las operaciones de un sistema de computación". Es la parte de un sistema que se puede codificar para ejecutarse en una computadora como un conjunto de instrucciones, e incluye la documentación asociada necesaria para comprender, transformar y usar esa solución. Estos documentos describen la organización del sistema, explican al usuario cómo utilizarlo y obtener eventuales actualizaciones del producto.

El software es un producto intangible, de alto contenido intelectual, que no sufre desgaste alguno y que puede ser potencialmente modificado de forma permanente. No se manufactura sino que se desarrolla a través de proyectos que son realizados por equipos de personas altamente formadas. A pesar de la tendencia a desarrollar componentes que puedan ser reutilizados y adaptados a diferentes necesidades, la gran mayoría del software se construye a medida.

El proceso de desarrollo de software es un conjunto de actividades que da como resultado un producto que responde a las necesidades de un usuario. Este proceso define todas aquellas actividades necesarias para transformar los requerimientos de un usuario en un producto. Estas actividades se las puede organizar con diferentes criterios y, cómo es un proceso intelectual-creativo, depende de las personas que formen parte del equipo y de sus decisiones y juicios. Existen varios procesos o modelos de desarrollo y podríamos catalogarlos en Secuenciales o Definidos, y en Empíricos. Los primeros responden siempre de la misma forma ante una determinada entrada, mientras que los segundos dependen de la evolución y adaptación de etapas anteriores.

Proceso Definido o Secuencial



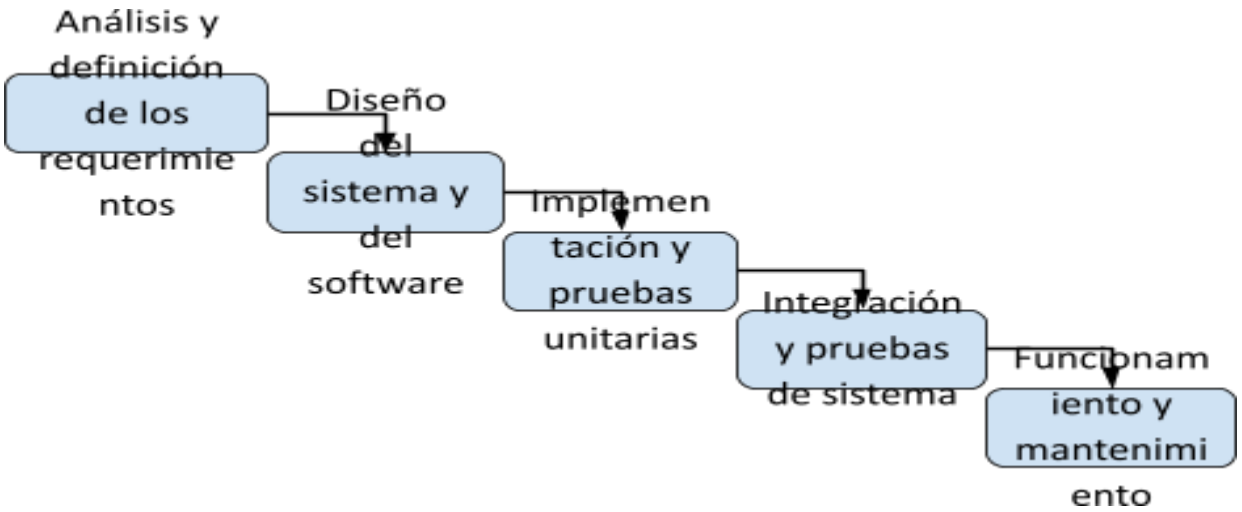
El conjunto de fases (o procesos) por las que pasa el software desde que se concibe y se desarrolla hasta que finaliza su uso (retiro de servicio) se conoce como ciclo de vida del software. Estas fases están estandarizadas, existiendo un marco de referencia que contiene los procesos, las actividades y las tareas involucradas en el desarrollo, explotación y mantenimiento de un producto software, abarcando la vida del sistema, desde la definición de sus requerimientos hasta la finalización de su uso.

Metodología en Cascada

El modelo de desarrollo en cascada fue presentado por primera vez en 1970 por Royce. Se lo conoce también como modelo lineal secuencial. Su principal característica es que cada fase del desarrollo está bien definida y separada, siguiendo un orden secuencial en el que cada etapa depende de la finalización de la anterior y que esta última haya pasado un proceso de validación que apruebe el paso a la siguiente.

Comúnmente las etapas del modelo son:

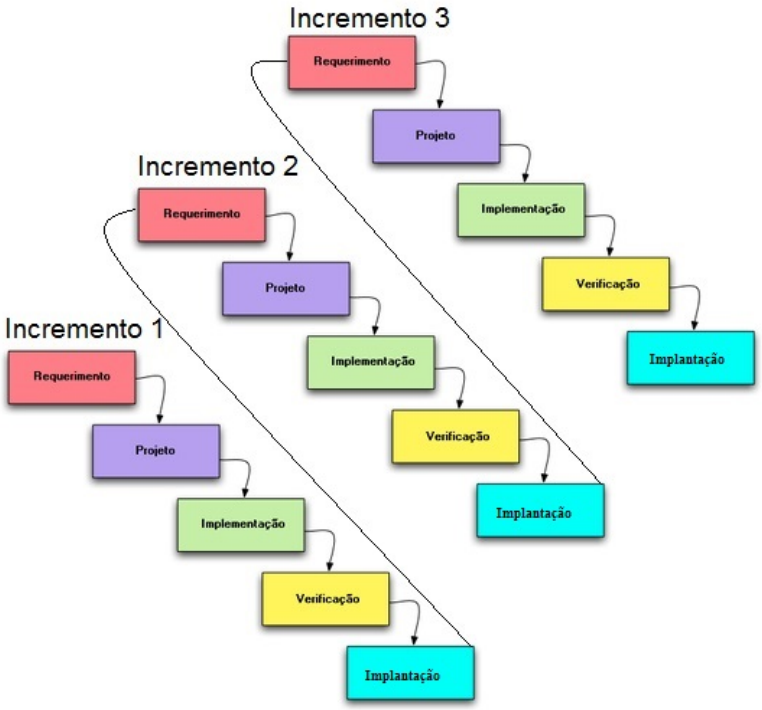
1. Análisis y definición de los requerimientos.
2. Diseño del sistema y del software.
3. Implementación y pruebas unitarias.
4. Integración y pruebas de sistema.
5. Funcionamiento y mantenimiento.



Ingeniería del software. Sommerville, I. Capítulo 4.

Metodología Iterativa e Incremental

La metodología iterativa e incremental deriva del proceso de desarrollo en cascada pero, con la diferencia de que aquí se admite que las etapas se solapan en tiempo con la finalidad de flexibilizar el tiempo de desarrollo total y así poder alcanzar resultados funcionales de manera temprana.



Metodología de desarrollo iterativo y creciente. Modelado Y Gestión De La Información (Blog http://modelado-de-la-informacion.blogspot.com/2015/09/metodologia-de-desarrollo-iterativo-y.html)

Esta metodología involucra dos procesos fundamentales:

- El proceso **incremental**: con esto se busca desarrollar una parte del producto que se pueda integrar al conjunto a medida que se alcanza un grado de completitud.
- El proceso **iterativo**: se realiza en ciclos donde se revisa y mejora el producto. De manera que la calidad del producto aumente, y no siempre implica la integración de nuevas funcionalidades.

Procesos Ágiles

Los métodos o procesos ágiles consideran un enfoque iterativo para las etapas de especificación, desarrollo y entrega del software. Estos métodos fueron pensados para el desarrollo de aplicaciones donde los requerimientos del sistema cambian rápidamente y también están enfocados en poder entregar software funcional de forma rápida a los

usuarios, quienes pueden evaluar y proponer nuevos requerimientos o cambios para iteraciones próximas.

Estos procesos se basan en los valores y los principios definidos en el Manifiesto Ágil elaborado en 2001 por un grupo de profesionales críticos de los modelos de desarrollo de software propuestos hasta el momento a los que consideraban excesivamente pesados y rígidos por su carácter normativo y fuerte dependencia de planificaciones detalladas previas al desarrollo.

Los valores que propone este manifiesto son:

- Valorar más a los individuos y sus interacciones que a los procesos y las herramientas
- Valorar más el software funcionando que la documentación exhaustiva
- Valorar más la colaboración con el cliente que la negociación contractual
- Valorar más la respuesta ante el cambio que seguir un plan

También se establecen los siguientes 12 principios:

1. Nuestra principal prioridad es satisfacer al cliente a través de la entrega temprana y continua de software de valor.
2. Son bienvenidos los requisitos cambiantes, incluso si llegan tarde al desarrollo. Los procesos ágiles se doblegan al cambio como ventaja competitiva para el cliente.
3. Entregar con frecuencia software que funcione, en periodos de un par de semanas hasta un par de meses, con preferencia en los períodos breves.
4. Las personas del negocio y los desarrolladores deben trabajar juntos de forma cotidiana a través del proyecto.
5. Construcción de proyectos en torno a individuos motivados, dándoles la oportunidad y el respaldo que necesitan y procurándoles confianza para que realicen la tarea.
6. La forma más eficiente y efectiva de comunicar información de ida y vuelta dentro de un equipo de desarrollo es mediante la conversación cara a cara.
7. El software que funciona es la principal medida del progreso.
8. Los procesos ágiles promueven el desarrollo sostenido. Los patrocinadores, desarrolladores y usuarios deben mantener un ritmo constante de forma indefinida.
9. La atención continua a la excelencia técnica enaltece la agilidad.
10. La simplicidad como arte de maximizar la cantidad de trabajo que no se hace, es esencial.
11. Las mejores arquitecturas, requisitos y diseños emergen de equipos que se autoorganizan.
12. En intervalos regulares, el equipo reflexiona sobre la forma de ser más efectivo y ajusta su conducta en consecuencia.

Si bien hay diferentes métodos ágiles, todos comparten algunos principios elementales:

- Participación del cliente: Los clientes deben estar implicados en todo el proceso de desarrollo, su rol está enfocado en proporcionar y dar prioridad a nuevos requerimientos del sistema, como así también participar en la evaluación de los entregables de cada iteración.
- Entrega incremental: El software se desarrolla en incrementos y, el cliente es quién determina qué requerimientos incluir en cada incremento.
- Enfocado en personas, no en procesos: Hay que identificar y explotar las habilidades de cada una de las personas del equipo. Cada equipo debe desarrollar sus propias formas de trabajar, es decir, sin imponer procesos formales.
- Aceptar el cambio: Siempre se debe tener en cuenta que los requerimientos del sistema pueden cambiar, y por lo tanto el diseño debe contemplar esto.
- Mantener la simplicidad: Se debe enfocar la simplicidad tanto en el software que se desarrolla como en el proceso de desarrollo.

Existen diferentes metodologías ágiles que si bien comparten los mismos principios, cada una posee sus propias características que las diferencian entre sí. Algunas de ellas son la Programación Extrema (XP, Extreme Programming), el Desarrollo Rápido de Aplicaciones, el Desarrollo de prototipo, y Scrum.

Scrum

Scrum es un marco de trabajo a través del cual las personas pueden abordar problemas complejos adaptativos, a la vez que se entregan productos de forma eficiente y creativa con el máximo valor. En scrum se aplican un conjunto de buenas prácticas para trabajar de manera colaborativa y obtener así el mejor resultado posible en proyectos complejos que demanden constantes cambios y adaptaciones.

La palabra scrum tiene su origen en un ámbito alejado al de la gestión de proyectos: el deporte. En rugby, “Scrum” es el término que define a la formación en la que ambos equipos empujan de forma coordinada para obtener la pelota. Esta formación requiere a cada equipo empujar no sólo en la misma dirección sino al mismo tiempo para lograr la suma de las fuerzas individuales. Representa una claro ejemplo de trabajo en equipo y coordinación entre todos los miembros de un mismo equipo.

Uno de los pilares de Scrum es la transparencia. Los aspectos significativos del proceso deben ser visibles para todos aquellos que son responsables del resultado. Esto requiere que dichos aspectos sean definidos en base a un estándar común, de tal modo que los observadores compartan un entendimiento común de lo que están viendo. Por ejemplo:

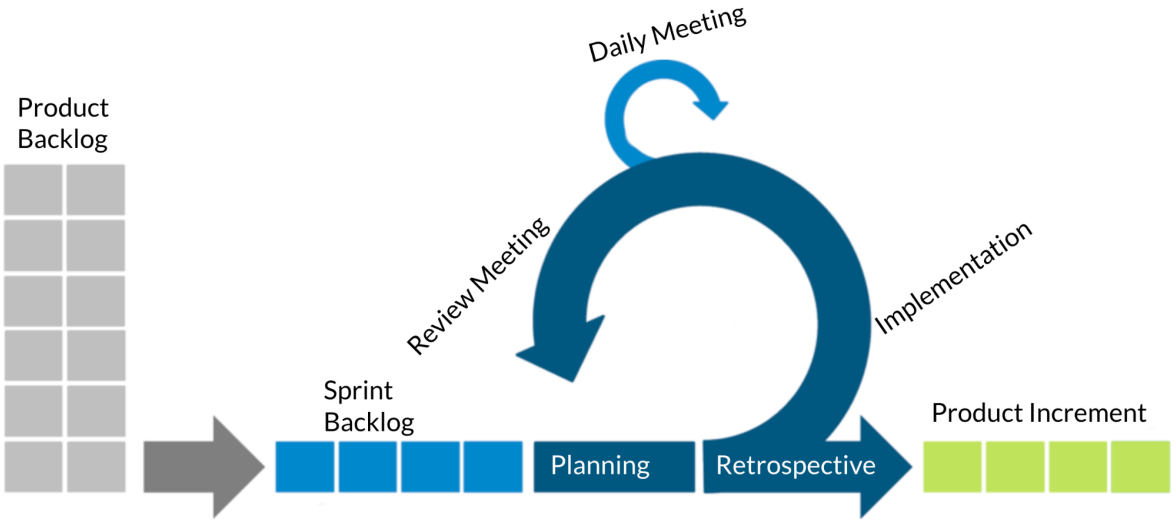
- Deben compartir un lenguaje común todos los participantes para referirse al proceso; y,
- Aquellos que desempeñan el trabajo y quienes inspeccionan el incremento resultante deben compartir una definición común de “Terminado” (“Done”).

Otro de los pilares es la inspección. Los usuarios de Scrum deben inspeccionar frecuentemente los artefactos de Scrum y el progreso hacia un objetivo para detectar variaciones indeseadas. Su inspección no debe ser tan frecuente como para que pueda interferir en el trabajo. Las inspecciones son más beneficiosas cuando se realizan de forma diligente por inspectores expertos en el mismo lugar de trabajo.

Finalmente, otro pilar de la metodología es su adaptación. Si un inspector determina que uno o más aspectos de un proceso se desvían de los límites aceptables y que el producto resultante será inaceptable, el proceso o el material que está siendo procesado deben ajustarse. Dicho ajuste deberá realizarse cuanto antes para minimizar desviaciones mayores.

Los proyectos se llevan a cabo en equipos que trabajan en ciclos temporales cortos y de duración fija. Cada ciclo se conoce como Sprint y tiene que proporcionar un resultado completo, un incremento del producto final que sea susceptible de ser entregado con el mínimo esfuerzo al cliente cuando este lo solicite. Los componentes del ciclo son:

- Eventos: Planificación del sprint, Scrum diario, Revisión del sprint, y Retrospectiva del sprint
- Roles: Desarrollador, Propietario del producto, y Scrum master
- Artefactos: Pila del producto, Pila del sprint, e Incremento.



Ciclo de desarrollo (Sprint) siguiendo metodología Scrum

Eventos

El núcleo de esta metodología son los Eventos. Cada uno de estos, se enmarca en el ciclo iterativo denominado Sprint y hace que sea posible llegar a un entregable útil al final de cada iteración. Los *Sprints* son el corazón de la metodología. Son períodos de tiempo fijos, generalmente de un mes o menos, y comienzan inmediatamente después de finalizado el *Sprint* anterior. Durante un *Sprint* no se pueden hacer cambios que perjudiquen los objetivos del mismo, se refina la pila de producto y eventualmente se podría clarificar o renegociar el alcance con el propietario del producto a medida que se aprende sobre el avance. Únicamente si el objetivo se vuelve obsoleto, el propietario del producto tiene la autoridad para cancelar el *Sprint*.

Planificación de Sprint

La planificación (Planning) es el evento que da inicio al *Sprint* y es donde se lleva a cabo la organización del trabajo a realizarse. Este evento no es necesariamente exclusivo del equipo de *Scrum*, sino que se puede invitar a otras personas para que aconsejen en los temas pendientes.
Durante la planificación se tratan los siguientes temas:

- ¿Por qué es valioso este *Sprint*?
  El propietario del producto propone como el producto va a aumentar su valor y utilidad con el *Sprint*. Todo el equipo colabora para definir el objetivo del *Sprint* que transmite el porque es valioso para los interesados en el producto. Al final de este evento el objetivo debe estar bien definido.
- ¿Qué se puede hacer en este *Sprint*?
  A lo largo de la discusión, los desarrolladores eligen ítems de la pila de producto, para incluirlos en el *Sprint* en curso. Durante este proceso, el equipo puede refinar estos ítems mejorando el entendimiento sobre cada uno. Este tema, de conocer cuánto se puede hacer, los desarrolladores lo entrenan teniendo en cuenta su rendimiento pasado, la capacidad y su definición de Terminado.
- ¿Cómo se llevará a cabo el trabajo seleccionado?
  Para cada uno de los ítems seleccionados, los desarrolladores planean el trabajo necesario para crear un incremento que cumpla con la definición de Terminado. La manera en que estos lo ejecutan queda a su discreción y se suele descomponer los ítems de la pila de producto en tareas más pequeñas, de un día o menos.

Scrum Diaria

Las *Scrum* diarias (Daily Scrum) son reuniones de no más de 15 minutos, en las cuales se sincroniza el trabajo desarrollado y se establece el plan para el día en curso. Generalmente se recomienda hacerlas viendo el tablero con la pila de sprint. En las reuniones participan los

desarrolladores, pero también pueden participar otras personas relacionadas con el proyecto o la organización, aunque normalmente estas no intervienen. Se comparten, los avances, los problemas y las ideas, también si es necesario se actualiza la pila de *Sprint*. Es importante tener en cuenta que no es una reunión de inspección o control, sino que de comunicación entre el equipo.

Revisión de Sprint

Sobre el final del *Sprint* se hacen las reuniones de revisión (Review Meeting) de *Sprint* para comprobar el incremento. Estas suelen durar entre una y dos horas, en caso de que el incremento tenga una complejidad muy alta, podrían extenderse aún más. Estas reuniones permiten al equipo y al propietario del producto tomar sensibilidad del ritmo de construcción y la trayectoria que cobra la visión del producto. Son instancias en las que se reajusta la pila de producto para incluir nuevas oportunidades.

Retrospectiva de Sprint

El propósito de las reuniones de Retrospectiva (Retrospective) de *Sprint* es planear nuevas formas de mejorar la calidad y efectividad. Se analiza lo que sucedió en la última Sprint con respecto a los individuos, sus interacciones, los procesos, lo que fue bien y los problemas que se encontraron, los que se resolvieron y los que quedaron pendientes.

Roles / Equipo

La unidad fundamental de Scrum es un equipo de personas (*Scrum Team*). Cada equipo consiste en un *Scrum Master*, un propietario del producto (*Product Owner*) y desarrolladores (*Developers*).
El equipo debe ser suficientemente pequeño como para ser ágil y grande como para completar una cantidad considerable de trabajo dentro de un *Sprint*. Normalmente los equipos están conformados por 10 personas o menos, ya que en general, los equipos pequeños se comunican mejor y son más productivos.
El equipo es responsable de todas las actividades relacionadas al producto desde las colaboración con las partes interesadas, verificación, mantenimiento, experimentación, desarrollo y cualquier otra cosa que pueda ser requerida.

Desarrolladores

Los desarrolladores son aquellos que producen cada uno de los incrementos de cada *sprint* y siempre son de responsables de:

- Crear un plan para el *Sprint*, la pila de *Sprint* (*Sprint Backlog*).
- Asegurar la calidad mediante la adopción de una Definición de Terminado.
- Adaptar el plan de cada día para conseguir el Objetivo del *Sprint* (*Sprint Goal*).

Propietario del Producto

El propietario del producto (*Product Owner*) es el responsable de gestionar efectivamente la Pila de Producto (*Product Backlog*), lo que incluye:

- Desarrollar y comunicar el objetivo del producto (*Product Goal*).
- Crear y comunicar cada uno de los ítems de la pila de producto.
- Ordenar los ítems de la pila de producto siguiendo alguna prioridad.
- Asegurar que la pila de producto sea transparente, visible y comprensible.

Es importante resaltar que el propietario del producto es una única persona, y por tanto, se deben respetar sus decisiones. Estas, deben ser reflejadas en el contenido y orden en la pila de producto, como así también en el incremento y la revisión del *Sprint*.

Scrum Master

El *Scrum Master* es el encargado de asegurar que la metodología se lleve a cabo tal y como está definida. Esto se logra haciendo que cada persona del equipo entienda la teoría y práctica de la metodología.

La persona con el rol de *Scrum Master* sirve al equipo de Scrum de varias formas:

- Orienta a los miembros del equipo en la autogestión y la multifuncionalidad.
- Ayuda a los equipos a enfocarse en crear incrementos de alto valor que cumplan con la definición de Terminado.
- Remueve los impedimentos de progreso que pudiera tener el equipo.
- Asegura que todos los eventos de *Scrum* se lleven a cabo y que sean positivos, productivos y dentro de los tiempos establecidos.

El *Scrum Master* también colabora con el propietario del producto de las siguientes maneras:

- Ayuda a encontrar técnicas para definir efectivamente el objetivo del producto y la gestión de la pila de producto.
- Ayuda a que los miembros del equipo entiendan los items de la pila de producto.
- Ayuda a establecer un planeamiento del producto de manera empírica para entornos que sean complejos.

Finalmente también colabora con el resto de la organización con actividades como:

- Liderar, planear y entrenar a los miembros de la organización en lo que respecta a la adopción de *Scrum*.
- Planear y aconsejar como implementar *Scrum* dentro de la organización.
- Ayudar a los empleados a comprender y llevar a la práctica la metodología, mediante un enfoque empírico para trabajos complejos.

Durante la reunión de planificación de *Sprint*, el *Scrum Master* deberá desempeñar las siguientes funciones:

- Realizar esta reunión antes de cada *Sprint*.

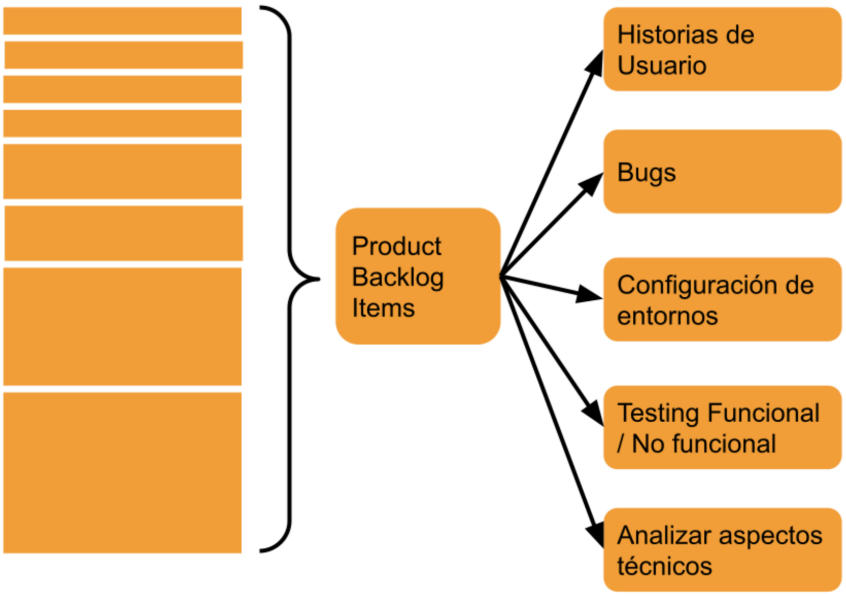
- Asegurar que se cuenta con una pila de producto preparada por el propietario del producto.
- Ayudar a mantener el diálogo entre el propietario del producto y los desarrolladores.
- Asegurar que se llegue a un acuerdo entre el propietario del producto y los desarrolladores con respecto a lo que incluirá el incremento.
- Ayudar a comprender la visión y necesidades de negocio del cliente.
- Asegurar que se ha realizado una descomposición y estimación del trabajo realistas.
- Asegurar que al final de la reunión estén determinados los siguientes puntos:
    - Los ítems de la pila de producto que se van a ejecutar
    - el objetivo del *Sprint*.
    - La pila de *Sprint* con todas las tareas estimadas.
    - La duración del *Sprint* y la fecha de reunión de revisión.
    - La definición de Terminado que determinará que el incremento está listo.

Artefactos

Pila de Producto

La pila de producto (*Product Backlog*) es una lista ordenada de lo que se necesita para mejorar el producto. Cada una de las entradas de esta lista son potenciales trabajos a seleccionarse para su realización durante una reunión de planificación de *Sprint*. El refinamiento de los ítems de la pila de producto es el acto de convertir esos ítems en elementos más detallados y precisos, en este proceso también se les asigna una prioridad y un “tamaño”.

La pila de producto puede incluir ítems para explorar las necesidades del cliente, analizar opciones técnicas, y otros ítems de trabajo tales como la corrección de errores (bugs) o la configuración del entorno. Todo lo que esté en la pila de producto representa o aporta a conseguir el objetivo del producto (*Product Goal*) y este último es el objetivo que el equipo tiene que conseguir cumplir a largo plazo.



Tipos de elementos en la pila de producto

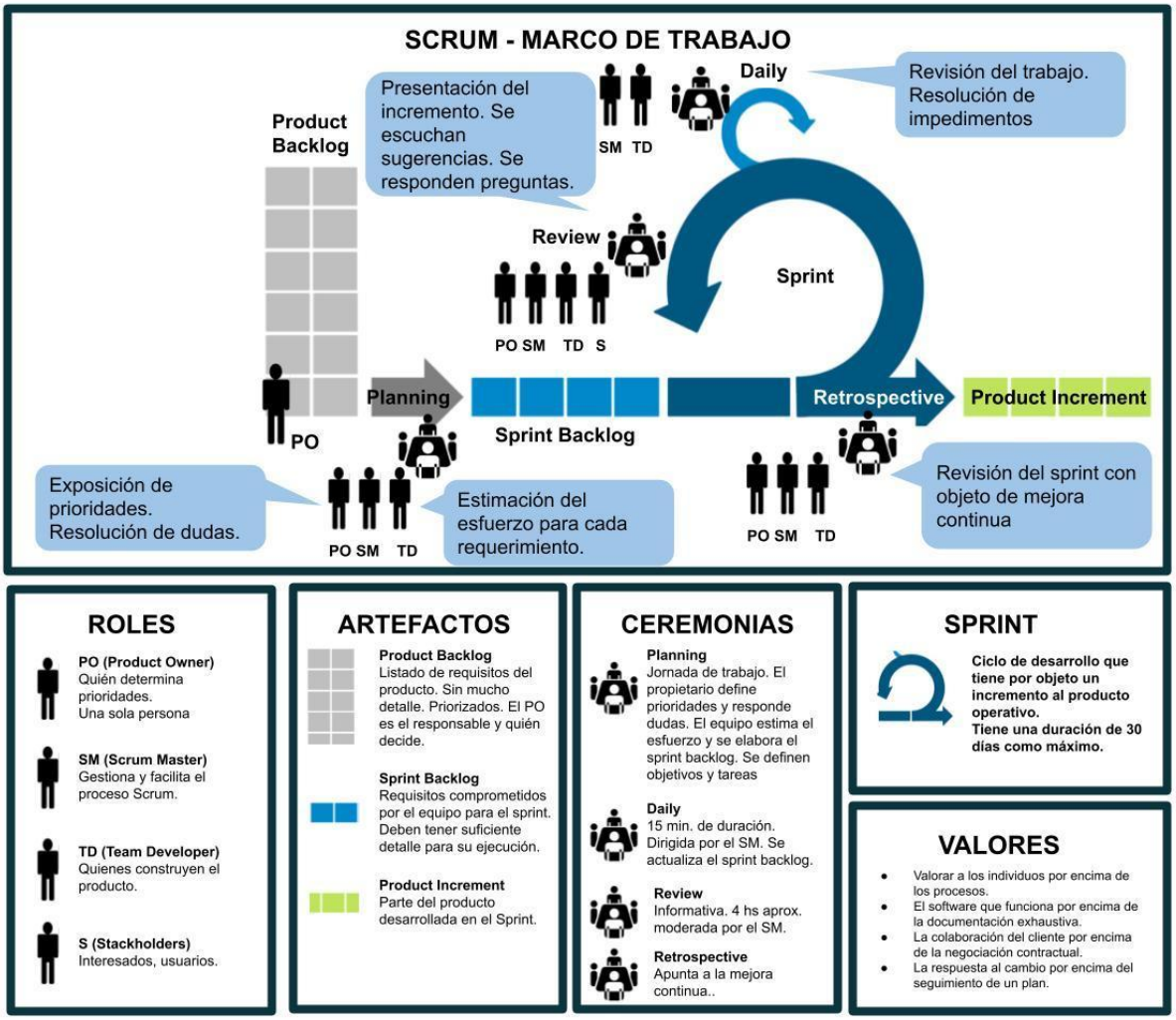
Pila de Sprint

Este artefacto, pila de *Sprint* (*Sprint Backlog*), delimita el trabajo necesario para alcanzar determinado incremento, y sirve para marcar el avance. También sirve como herramienta de comunicación del equipo. Es visible para todos y representa una imagen en tiempo real del trabajo que llevan los desarrolladores para alcanzar el objetivo del *Sprint* (*Sprint Goal*). Este último se crea durante la reunión de planificación de *Sprint* y, a lo largo de todo el ciclo, los desarrolladores lo tienen presente.

Incremento

Un incremento de producto (Product Increment) es un paso concreto que acerca el desarrollo al objetivo del producto. Cada incremento agrega valor al incremento anterior y, bajo un proceso de verificación, se asegura que todos los incrementos trabajen bien juntos.

Existe un compromiso entre el incremento y la definición de terminado (*Definition of Done*). Esta definición es una descripción formal del estado que debe alcanzar el incremento para cumplir con el nivel de calidad requerido para el producto. Si por alguna razón algún ítem de la pila de producto no cumple con la definición de terminado, no debe formar parte del entregable y ni siquiera ser presentado en la reunión de revisión de Sprint. En lugar de esto, debe volver a la pila de producto para ser considerado en un futuro.



Cuadro resumen de Scrum

Especificaciones y Requerimientos

La etapa de descubrir, analizar, documentar y verificar los servicios y restricciones del sistema, se la denomina ingeniería de requerimientos (*Requirements Engineering*). Es importante realizar una especificación de los requerimientos, es decir documentar de forma completa, precisa y verificable los requisitos, el diseño y el comportamiento u otras características de un sistema o componentes del mismo.

Requerimientos Funcionales

Los requerimientos funcionales son declaraciones de los servicios que debe proporcionar el sistema. Describen cómo debe reaccionar el sistema a entradas particulares o cómo debe comportarse bajo determinadas condiciones.

Por tanto, la especificación de requerimientos debería cumplir las características de ser completa y consistente. Para que sea completa, todos los servicios descritos por el usuario deben estar definidos. Por otra parte, para que sea consistente los requerimientos no deben ser contradictorios.

Requerimientos no Funcionales

Los requerimientos no funcionales son aquellos que se refieren a las propiedades que debe tener, como fiabilidad, capacidad de almacenamiento, tiempo de respuesta, etc.
Generalmente surgen de necesidades del usuario que tienen que ver con restricciones de presupuesto, políticas, la interoperabilidad con otros sistemas, factores externos, regulaciones, privacidad, seguridad, etc.

Requerimientos en Scrum y Estimación de tiempos

Dentro de las metodologías ágiles se suelen utilizar las historias de usuario (*User Stories*) como herramienta para definir los requerimientos del sistema. Estas son descripciones o especificaciones cortas de una funcionalidad, validada por un usuario o cliente del sistema.
Generalmente las historias se escriben en lenguaje que el usuario pueda entender y que refleje una descripción sintetizada de lo que este desea. En lo posible se debe tratar de eliminar ambigüedades y malas interpretaciones.
Para redactar una historia de usuario se tiene que seguir una estructura como la siguiente:

> Como ***<usuario>*** quiero/puedo ***<algun objetivo>*** para que/de forma que ***<motivo>***.

Escribir las historias de usuario con este formato tiene ciertas ventajas:

- Primera persona: Invita a quien la lee a ponerse en el lugar del usuario.
- Priorización: Ayuda al propietario del producto a priorizar y visualizar mejor cual es la funcionalidad, quién se beneficia y cuál es el valor.
- Propósito: la presencia del propósito de una funcionalidad brinda al equipo la posibilidad de plantear alternativas que cumplan con el mismo objetivo en caso que el costo de la funcionalidad sea demasiado alto o su construcción no sea viable.

Podemos redactar la historia de usuario del siguiente ejemplo: “Daniel llegó hoy a Neuquén pero está perdido en su automóvil y dio vueltas por horas. Quiere llegar al museo provincial de bellas artes “Museo Nacional de Bellas Artes de Neuquén”. En su celular deberá buscar como llegar si la calle es Mitre y Santa Cruz.” utilizando la estructura propuesta de la

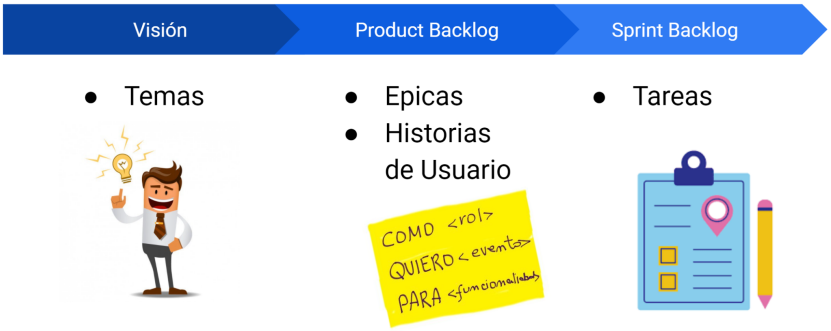
siguiente forma: Como *conductor* quiero *buscar un destino a partir de una calle y altura* para poder *llegar al lugar deseado*.

Temas, Épicas e Historias

Durante el proceso de análisis de requerimientos se suele agrupar a las historias de usuario en épicas y a su vez, estas últimas están relacionadas a un tema en particular.

Relación entre Temas, Épicas e Historias de usuario.

Usualmente las historias de usuario se las ubica en la pila del producto, son derivadas de los temas y agrupadas en épicas, posteriormente se traducen en tareas en la pila de *Sprint*.



Historias de Usuario en Scrum

Método INVEST

Este método, desarrollado por Bill Wake en 2003, permite asegurar la calidad de la escritura de las historias de usuario. Para esto debe cumplir con las siguientes características:

- Independiente (*Independent*): Cada historia de usuario puede ser planificada e implementada en cualquier orden. No dependen unas de otras, si esto ocurre se deben dividir o combinar.
- Negociable (*Negotiable*): Las historias deben ser negociables ya que sus detalles serán acordados por el cliente/usuario y el equipo durante la fase de «conversación».
- Valiosa (*Valuable*): Una historia de usuario tiene que ser valiosa para el cliente o el usuario.

- Estimable (*Estimable*): Una buena historia de usuario debe ser estimada con la precisión suficiente para ayudar al cliente o usuario a priorizar y planificar su implementación. Si no podemos estimarla debemos incidir en la fase de conversación o dividirla.
- Pequeña (*Small*): Pequeñas. Solemos hacerlas de tal modo que ocupen como máximo un sprint.
- Comprobable (*Testable*): La historia de usuario debe poderse probar (Hemos trabajado con anterioridad en la fase "confirmación" de la historia de usuario). Tanto el usuario como el equipo de desarrollo tienen que poder probarla para saber cuando está finalizada.

Método SMART

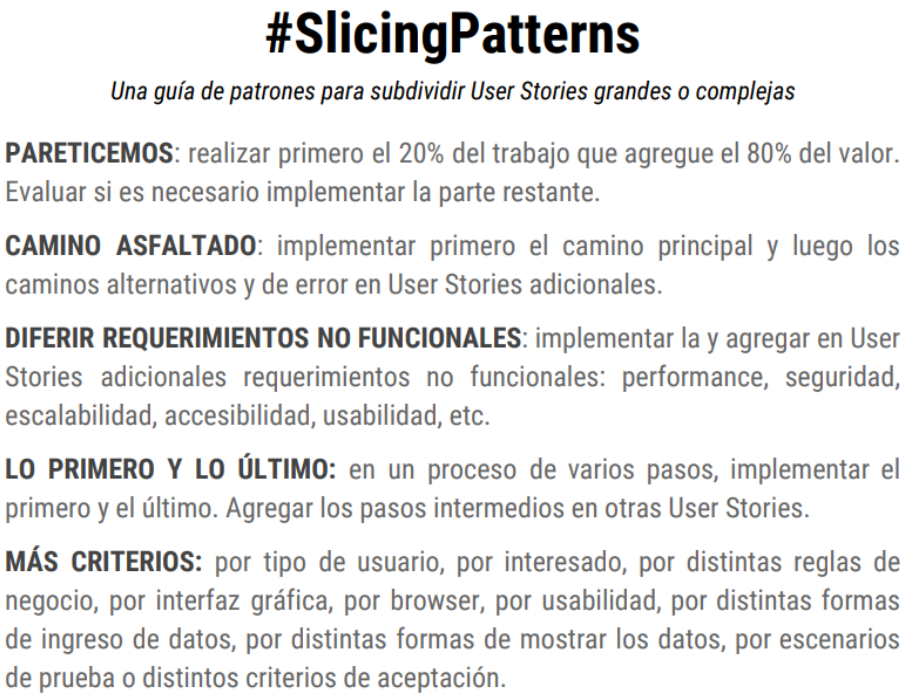
Cuando se descomponen las historias de usuario en tareas, se puede utilizar el método SMART, un método que sirve para evaluar si las tareas están correctamente definidas. SMART es un acrónimo (en inglés) de las siguientes características:

- Específico (*Specific*): Una tarea debe ser lo suficientemente específica para que todos puedan entenderla.
- Medible (*Measurable*): ¿Hace lo que se pretende? Es el equipo quien debe ponerse de acuerdo sobre lo que esto significa.
- Alcanzable (*Achievable*): ¿Es razonable la meta? Consejo: Es bueno establecer metas que representen un desafío, pero puede ser contraproducente no alcanzarlas.
- Relevante (*Relevant*): Cada tarea debe ser relevante, contribuyendo a la historia en cuestión.
- A tiempo (*Time-Boxed*): Una tarea debe estar limitada a una duración específica. No necesariamente debe ser una estimación formal en horas o días, pero debe haber una expectativa para que la gente sepa cuándo debe buscar ayuda.

Características del método SMART. Recuperado de: https://u.osu.edu/howze.10/2019/12/08/essential-to-achieve-ones-aspirations-smart-goals/

Subdivisión de Historias de Usuario

Existe una guía de patrones para subdividir historias de usuario grandes o complejas en otras más pequeñas y alcanzables.



#SlicingPatterns

Una guía de patrones para subdividir User Stories grandes o complejas

PARETICEMOS: realizar primero el 20% del trabajo que agregue el 80% del valor. Evaluar si es necesario implementar la parte restante.

CAMINO ASFALTADO: implementar primero el camino principal y luego los caminos alternativos y de error en User Stories adicionales.

DIFERIR REQUERIMIENTOS NO FUNCIONALES: implementar la y agregar en User Stories adicionales requerimientos no funcionales: performance, seguridad, escalabilidad, accesibilidad, usabilidad, etc.

LO PRIMERO Y LO ÚLTIMO: en un proceso de varios pasos, implementar el primero y el último. Agregar los pasos intermedios en otras User Stories.

MÁS CRITERIOS: por tipo de usuario, por interesado, por distintas reglas de negocio, por interfaz gráfica, por browser, por usabilidad, por distintas formas de ingreso de datos, por distintas formas de mostrar los datos, por escenarios de prueba o distintos criterios de aceptación.

Patrones de división de historias de usuario. Recuperado de: http://guiasagiles.org/#SlicingPatterns

Esta técnica de subdivisión de historias de usuario se puede llevar a cabo mediante juegos de cartas como el disponible en este enlace.

Las 3 C de las Historias de Usuario

Una historia de usuario está compuesta de tres elementos que son fundamentales. La primera y trivial es la **tarjeta** (*Card*), es donde se escribe la historia de manera clara y con la mínima ambigüedad. El segundo componente es la **conversación** (*Conversation*), es importante debatir y validar con el cliente y el equipo de desarrollo, esto por lo general se realiza en la reunión de planificación. Y finalmente la **confirmación** (*Confirmation*), es importante confirmar que todos los involucrados han comprendido los requisitos.

Puntos de historia

Extraído de referencia 6, páginas 55-56:

"El trabajo necesario para realizar un requisito o una historia de usuario no se puede prever de forma absoluta porque rara vez son realidades de una solución única. En el caso de que se

pudiera, por otra parte, la complejidad de la medición haría una métrica demasiado pesada para la gestión ágil.
No resulta posible estimar con precisión la cantidad de trabajo que hay en un requisito. En consecuencia, tampoco se puede saber con antelación cuánto tiempo exigirá, porque a la incertidumbre del trabajo se suman las inherentes al tiempo: no se puede estimar la cantidad o la calidad del trabajo que realiza una «persona media» por unidad de tiempo, porque son muy grandes las diferencias de unas personas a otras. Es más: la misma tarea realizada por la misma persona requerirá diferentes tiempos según las 56 circunstancias.
Por todas estas razones, al estimar de forma ágil, se prefiere emplear unidades relativas. En gestión ágil se suelen emplear «puntos» como unidad de trabajo, usando denominaciones como «puntos de historia». Cada organización, según sus circunstancias y su criterio, institucionaliza su métrica de trabajo, su «punto». Es el tamaño relativo de tareas que se suele emplear. Es importante que el significado y la forma de aplicar la métrica sea siempre la misma en las mediciones de la organización, y que sea conocida por todos. El tipo de «punto» dependerá de la organización. En un equipo de programación el punto puede ser equivalente a preparar una pantalla de login; para un equipo de diseño gráfico, la maquetación de un tríptico.
El «punto» ayuda, por un lado, a dimensionar la estimación de una tarea comparándola con una ya conocida, y por otro lado, a contrastar la dificultad que la tarea presenta para cada miembro del equipo según sus especialidades. Un ejemplo para ilustrar esto último podría ser el esfuerzo que cuesta freír un huevo. Si se estima cuántos «huevos fritos» costaría planchar una camisa, la respuesta dependerá de la persona. Alguien puede ser muy habilidoso friendo huevos, pero muy torpe para planchar camisas, y estimará que eso le costaría «8 huevos fritos»; es decir, «8 puntos». Alguien muy acostumbrado a las tareas domésticas, en cambio, podría estimar la tarea en «un punto» o «un huevo frito».

Ambos tienen razón: la cuestión es que la persona que estime sea la que va a realizar la tarea. Los «puntos de historia» suelen ser una unidad relativa o abstracta basada en algo con lo que el equipo esté muy familiarizado.
Por último, con esto se puede estimar la velocidad: en scrum, ésta es igual a la cantidad de trabajo realizado por el equipo en un sprint. Así, por ejemplo, una velocidad de 150 puntos indica que el equipo realiza 150 puntos de trabajo en cada sprint.
No obstante, al salir del marco estándar de scrum podemos encontrar sprints de diferentes duraciones. Cuando esto sucede, se puede expresar la velocidad en unidades de tiempo en lugar de por sprint. Es decir: «la velocidad media del equipo es de x puntos por semana».”

Los puntos de historia representan un valor que sólo será relevante para el equipo de scrum, y que se usará para estimar el tamaño de una tarea (también llamada ítem, en inglés) en el backlog. Permite al equipo determinar qué tareas pueden realizar en un sprint. El product owner (“propietario del producto” en su traducción literal) podrá ver, por los puntos de la

historia, qué tareas fueron relativamente complicadas de hacer por el equipo. Los puntos dan una idea del tamaño y el esfuerzo que se necesita para que las tareas sean realizadas.

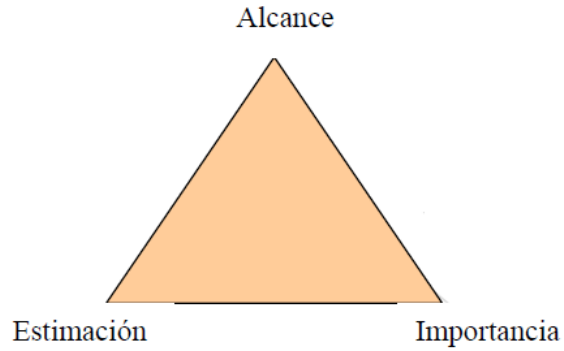
Para empezar con los puntos de la historia, el equipo debe primero definir una **historia de referencia (o pivote)** con la que después podrá comparar todas las demás historias. Se recomienda elegir una historia de usuario más o menos compleja, que incluya tantas disciplinas del equipo como sea posible. A esta historia le daremos un puntaje, posiblemente 5 u 8. Luego, se utiliza el pivote durante los perfeccionamientos del sprint backlog para determinar si otra historia de usuario es más pequeña o más grande, y qué valor puede darle.

Planificación

Extraído de referencia 10, Capítulo 3: ¿Cómo hacemos la planificación de Sprint?
"La planificación de Sprint es una reunión crítica, probablemente la más importante de Scrum. Una planificación de Sprint mal ejecutada puede arruinar por completo todo el Sprint. El propósito de la planificación de Sprint es proporcionar al equipo suficiente información como para que puedan trabajar en paz y sin interrupciones durante unas pocas semanas, y para ofrecer al Dueño de Producto suficiente confianza como para permitírselo.
Una planificación de Sprint produce, concretamente:

- *Una meta de Sprint.*
- *Una lista de miembros (y su nivel de dedicación, si no es del 100%)*
- *Una Pila de Sprint (lista de historias incluidas en el Sprint)*
- *Una fecha concreta para la Demo del Sprint.*
- *Un lugar y momento definidos para el Scrum Diario.*

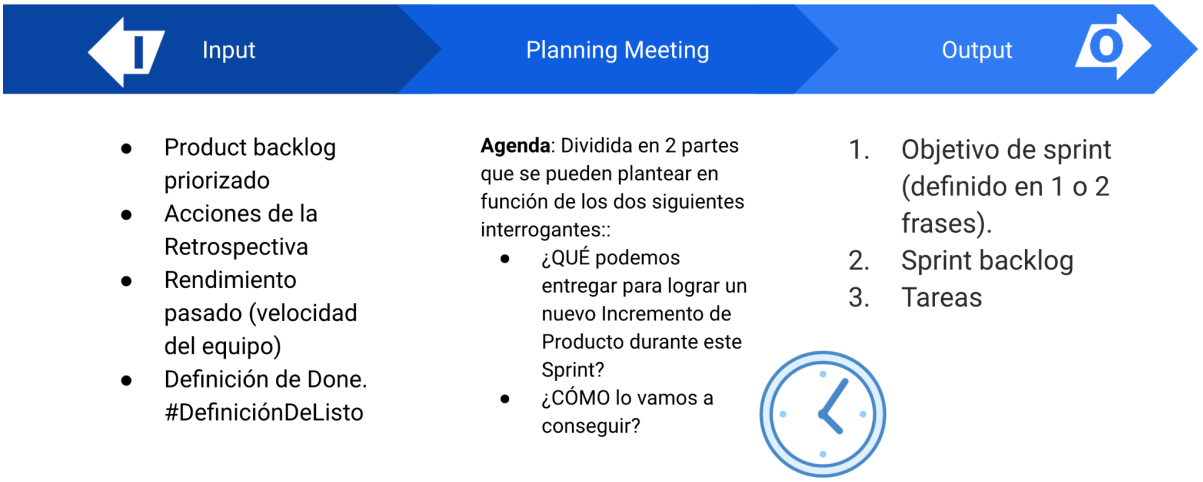
Es importante que tanto el equipo como el propietario del producto asistan a la planificación de Sprint ya que cada historia contiene tres variables que son muy dependientes unas de otras.



Variables involucradas en cada historia de un Sprint. Recuperado de referencia 10.

El alcance y la importancia los fija el propietario del producto. La estimación la proporciona el equipo. Durante una planificación de Sprint, estas variables sufren un ajuste fino y continuo a través del diálogo cara a cara entre el equipo y el propietario del producto."

La reunión que se realiza al comienzo de cada Sprint donde participa el equipo completo; sirve para inspeccionar el Product Backlog y que el equipo de desarrollo seleccione del Product Backlog Items en los que va a trabajar durante el siguiente Sprint. Durante esta reunión, el Propietario del producto presenta el Product Backlog actualizado que el equipo de desarrollo se encarga de estimar, además de intentar clarificar aquellos ítems que crea necesarios.
Durante esta etapa se inspeccionan el Product Backlog, los acuerdos de la Retrospectiva, la capacidad y la Definition of Done y se adaptan el Sprint Backlog, Sprint Goal y el plan para poder alcanzar ese Sprint Goal.



Entradas, organización y resultados de la reunión de planificación de *Sprint*.

Planning Poker

A la hora de estimar tiempos se trata de alcanzar los siguientes objetivos:

- Tener diferentes puntos de vista. En la estimación ágil se busca que todo el mundo participe y diga la suya. Si hay discusión mejor, ya que de lo que se trata es de avanzar todo lo posible los problemas y tenerlos en cuenta desde el inicio. No debería haber personas con una voz y voto más fuertes que los demás.
- Detectar posibles tareas ocultas y posibles obstáculos. La sesión de estimación es una de las primeras oportunidades de detectar riesgos que pueden comenzar a tratarse para que no se conviertan en impedimentos.
- Tener una visión compartida del trabajo que se va a realizar. Es muy útil haber participado en las estimaciones para después hacer las planificaciones. Conocer el

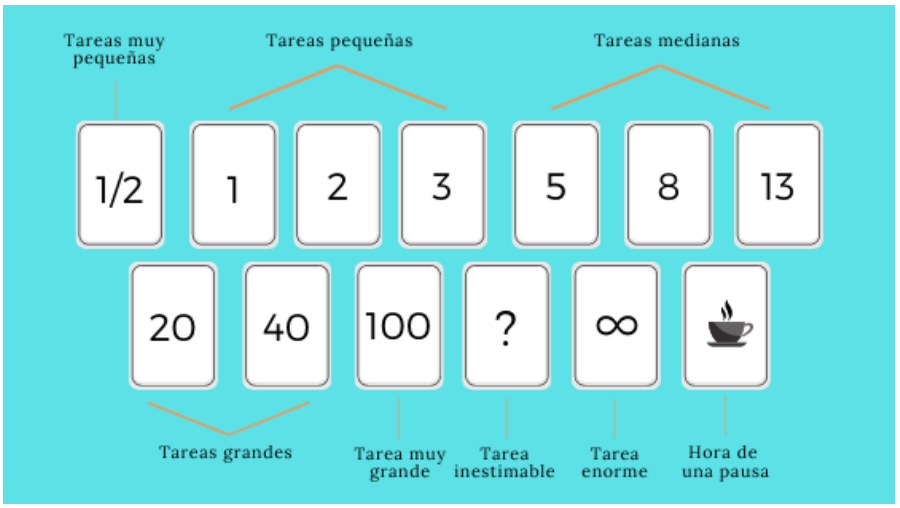
tamaño de las historias y tareas permite que sea más fácil comprometerse con un plan de trabajo.

- Tener estimaciones más realistas (no más precisas). Para ello necesitas eliminar la presión contractual, esto es, dejar margen para equivocarse y evitar así introducir buffers "inconscientes" por si acaso. Lo que buscamos es realismo, no precisión, es decir, quiero saber si una historia serán 3 o 5 días, si me dices que tardarás 26,5 horas dudaré de que hayas hecho un buen ejercicio de estimación.

Planning Poker es una técnica efectiva para la estimación ágil. Se reúne al equipo y se utiliza una baraja de poker modificada con la que se hacen rondas de estimación con ayuda de estas cartas.

La baraja de cartas tiene una pseudo secuencia de Fibonacci modificada, cada participante tendrá cartas con valores 0, ½, 1, 2, 3, 5, 10, ? e infinito, donde el 0 significa que la historia ya está hecha o no requiere esfuerzo, el interrogante significa que falta información para estimar la tarea o historia, finalmente el infinito significa que el trabajo es demasiado grande y habría que subdividirla.

Para comenzar la sesión de estimación se suele realizar una ronda de preguntas para estimar para despejar cualquier tipo de duda sobre las historias que se van a estimar. Luego se leen y discuten una por una las historias y cada uno de los integrantes elige una carta en función del esfuerzo que prevé requerirá esa historia, es importante aclarar que solo se pueden elegir valores incluidos en la baraja. Finalmente, si no hay consenso, se abre discusión donde cada uno explica su elección, luego se repite la estimación en busca de un consenso y, si al final no hay consenso, se elige la media o el valor más alto.



Valores de las cartas que se usan en la baraja de Planning Poker. Recuperado de https://www.datocms-assets.com/17507/1606818245-planning-poker-cards.png

Esta técnica tiene varias ventajas:

- Todos los miembros del equipo expresan su opinión sin sentirse condicionados por el resto.

- Al ser conscientes del esfuerzo que supone, aumenta el grado de implicación de los componentes del equipo.
- Al sentirse partícipes, el grado de compromiso con el proyecto también aumenta.
- Hay más efectividad a la hora de estimar las fechas de entrega del proyecto.

Referencias

1. Sommerville, I. (2005). *Ingeniería del Software*. Pearson Educación.
2. Fundingsoft (2021). Fundamentos de Ingeniería de Software. Recuperado de: https://fundingsoft.wordpress.com/unidades/
3. Modelado de la Información (2015). Metodología de desarrollo iterativo y creciente. Recuperado de: http://modelado-de-la-informacion.blogspot.com/2015/09/metodologia-de-desarrollo-iterativo-y.html
4. Proyectos Ágiles (2021). ¿Qué es Scrum? Recuperado de: https://proyectosagiles.org/que-es-scrum/
5. Scrum Guides (2020). The Scrum Guide. Recuperado de: https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf
6. Scrum Manager (2020). Scrum Master. Recuperado de: https://scrummanager.net/files/scrum_master.pdf
7. ASP Gems (2021). Historias de usuario: aplicando INVEST. Recuperado de: https://aspgems.com/historias-de-usuario-aplicando-invest-parte-1/
8. Be Agile My Friend (2021). Las Mejores Historias de Usuario INVEST & SMART. Recuperado de: https://beagilemyfriend.com/historias-de-usuario-invest-smart/
9. Guías Ágiles (2020). SlicingPatterns. Recuperado de: http://guiasagiles.org/#SlicingPatterns
10. Git Books (2020). Scrum y XP desde las trincheras. Recuperado de: https://gonztirado.gitbooks.io/scrum-y-xp-desde-las-trincheras