Tema 1.- METODOLOGÍAS ÁGILES
--
INTRODUCCIÓN A SCRUM

Ingeniería del SW II

Construir software no es igual que construir productos físicos, como construir un edificio o un coche

- **En la ingeniería del software**, a diferencia de la arquitectura u otras ingenierías tradicionales**, no se puede separar tan claramente la fase de diseño de la fase de construcción.**
- Debemos tener muy claro que **es casi imposible cerrar un diseño a la primera para pasarlo a codificación sin tener que modificarlo**.

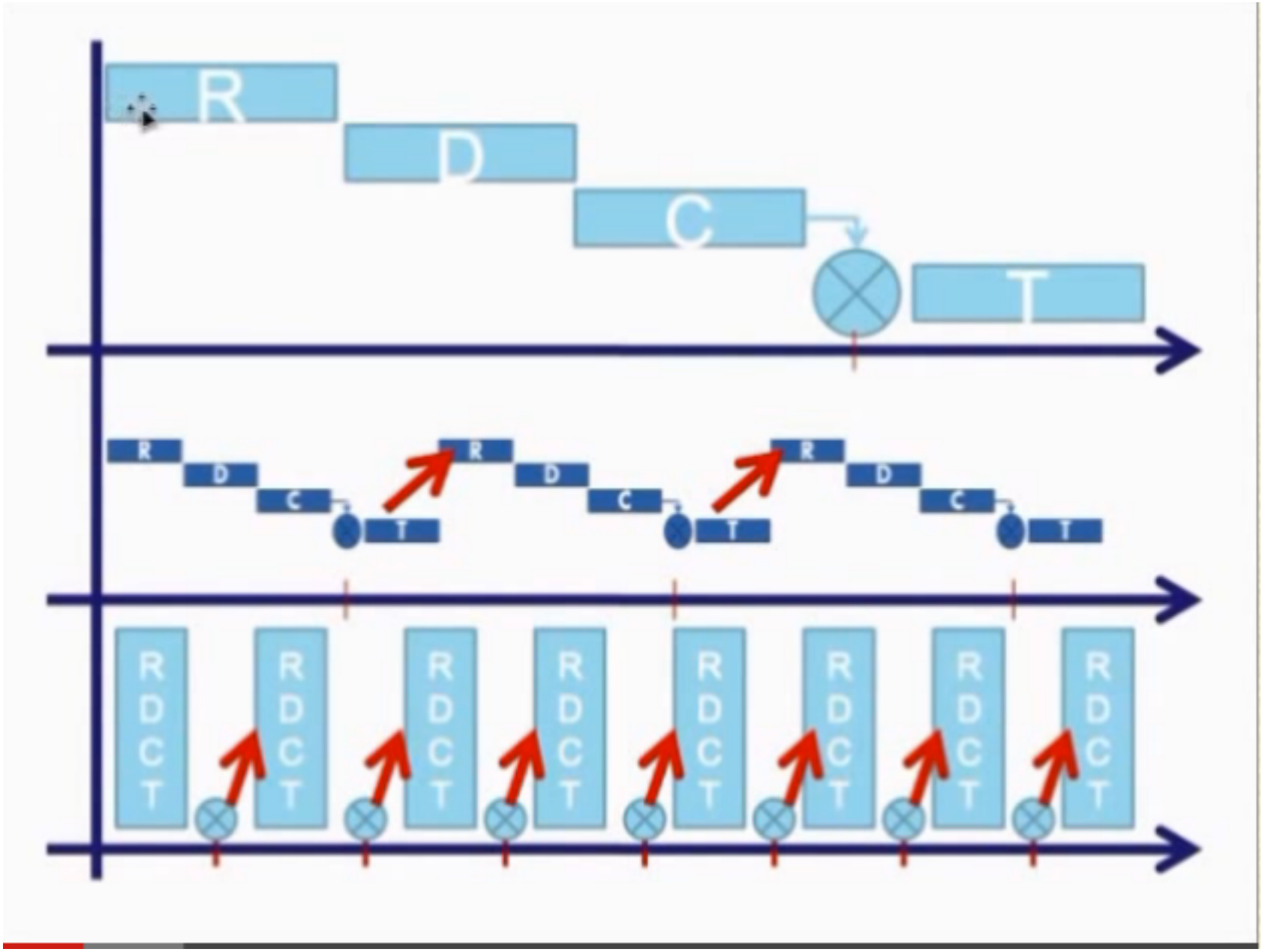
- Las **metodologías ágiles aceptan el cambio del software**, y están pensadas para convivir con esta constante evolución y cambio del producto software. **Cambiar el software no es un problema… es una ventaja.**

- **CONCLUSIÓN**
  - Intentar construir software de la misma forma que se hace un edificio puede implicar importantes problemas.
  - Los intentos por comparar e intentar emular la construcción de productos físicos han llevado a grandes errores en gestión de proyectos software (Ejemplos: añadir más programadores a un proyecto retrasado, medir la evolución de un proyecto por el nº de líneas codificadas….).

- Para optimizar los resultados del desarrollo de productos software surgen las METODOLOGÍAS ÁGILES:
  - Adecuadas para proyectos con **requisitos cambiantes**
  - Requieren **tiempos de desarrollo reducidos**
  - Exige mantener una **alta calidad**.

- Lectura recomendada:

http://www.javiergarzas.com/2011/10/diapositivas-evolucion-fabricacion-software.html


R
D
C
T
R
D
C
T
R
D
C
T
R
D
C
T
R D C T
R D C T
R D C T
R D C T
R D C T
R D C T
R D C T
R D C T

CICLO DE VIDA ITERATIVO E INCREMENTAL

- El ciclo de vida **iterativo e incremental** es una de las prácticas fundamentales de un proyecto ágil.
- En este tipo de ciclo de vida, se va liberando parte del producto (prototipos) periódicamente mediante iteraciones o ciclos de desarrollo.
- De la unión del ciclo de vida iterativo y el incremental **al final de cada iteración** se consigue **una versión estable del software**, añadiendo además nuevas funcionalidades a las versiones anteriores.

CICLO DE VIDA ITERATIVO E INCREMENTAL

- **El desarrollo incremental** surge para **eliminar los riesgos** asociados a construir productos software grandes o con alto grado de complejidad. Se centra en **desarrollar por partes** el producto software, para después **integrarlas** a medida que se completan. Un ejemplo de un desarrollo puramente incremental puede ser la agregación de módulos en diferentes fases.
- El **desarrollo iterativo** se centra en **revisar y mejorar** el producto ya creado. En cada ciclo se revisa y mejora el trabajo. Un ejemplo de desarrollo iterativo y no incremental es aquel basado en refactorizaciones en el que cada ciclo mejora más el producto.

Ciclo de vida Iterativo e Incremental:
VENTAJAS

- **Visión de avance en el desarrollo**, viendo trabajo operativo real desde etapas iniciales del ciclo .
- **Obtención del "feedback" del usuario sobre un prototipo operativo**. Así puede orientar el desarrollo hacia el cumplimiento de sus necesidades, y realizar todas las adaptaciones identificadas para cumplir con los objetivos planteados.
- **Permite manejar el riesgo del proyecto**, apuntando a la resolución de los problemas por partes.
- **El aprendizaje y experiencia del equipo iteración tras iteración** sobre un prototipo operativo, lo que mejora exponencialmente el trabajo, **aumenta la productividad** y permite **optimizar el proceso** en el corto plazo.

MANIFIESTO ÁGIL

http://agilemanifesto.org/iso/es/manifesto.html

- Tras los fracasos en el desarrollo de proyectos software por los métodos tradicionales surgen nuevas propuestas metodológicas que buscan desarrollos más rápidos sin disminuir la calidad.

- En 2001, en EEUU se reunieron 17 expertos de la industria del software para debatir sobre metodologías, principios y valores que deben regir el desarrollo para conseguir software:
  - de buena calidad
  - en tiempos cortos
  - y flexible a los cambios.

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

MANIFIESTO ÁGIL: VALORES

http://agilemanifesto.org/iso/es/manifesto.html

- **"Valorar a los individuos e interacciones del equipo de desarrollo sobre el proceso y las herramientas."** Valoran el recurso humano como el principal factor de éxito. Se tendrán en cuenta las buenas prácticas de desarrollo y gestión de los participantes del proyecto (dentro del marco de la metodología elegida). Reconocen que es más importante construir un buen equipo de trabajo que las herramientas y procesos. El proceso se adapta a las necesidades de las personas y del equipo y no al revés.

- **"Software funcionando sobre documentación exhaustiva"**. No es necesario producir documentos a menos que sean necesarios de forma inmediata para tomar una decisión importante. Los documentos deben ser cortos y centrarse en lo fundamental. El problema no es la documentación sino su **utilidad**.

sobre= se valora más que

MANIFIESTO ÁGIL: VALORES

http://agilemanifesto.org/iso/es/manifesto.html

- **"Colaboración con el cliente sobre negociación contractual"**. Es necesaria una interacción constante entre el cliente y el equipo de desarrollo. Si el cliente no está disponible se recomienda tener una persona desde dentro de la empresa que represente al cliente, haciendo de interlocutor y participando en las reuniones del equipo.
- **"La respuesta al cambio sobre seguir estrictamente un plan"**. La planificación no debe ser estricta sino **adaptable y flexible**. Hay que tener en cuenta que el cambio existe y es muy importante adaptarse a él. Estos cambios pueden ir desde ajustes sencillos en la personalización del software hasta cambios en la legislación (que exijan formas diferentes de hacer las cosas), la aparición de nuevos productos en el mercado, comportamiento de la competencia, nuevas tendencias tecnológicas, etc

sobre= se valora más que

MANIFIESTO ÁGIL: PRINCIPIOS

1. **Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software que le aporte valor.** Se debe empezar a entregar software funcionando y útil al cliente en pocas semanas de forma que el cliente participe en el proceso probando, revisando y aprobando el software desde el principio.
2. **Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente**.
3. **Entregamos software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible**. Probar sw funcionando es más motivador para el cliente que simples informes, modelos abstractos y planes.
4. **Los responsables de negocio y los desarrolladores trabajamos juntos de forma cotidiana durante todo el proyecto**. El usuario debe estar involucrado todo el tiempo que dure el proyecto.

MANIFIESTO ÁGIL: PRINCIPIOS

5. **Los proyectos se desarrollan entorno a individuos motivados.** Hay que darles el entorno y el apoyo que necesitan, y confiarles la ejecución del trabajo.
6. **El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara**. El trabajo en equipo debe apoyarse en un buen sistema de comunicación entre los miembros del equipo y con el usuario.
7. **El software funcionando es la medida principal de progreso**. La cantidad de requisitos implementados y aprobados por el cliente es la mejor medida del avance del proyecto.
8. **Los procesos ágiles promueven el desarrollo sostenible**. Los promotores, desarrolladores y usuarios deben ser capaces de **mantener un ritmo constante** de forma indefinida. (Desde el principio asignar responsabilidades que se puedan cumplir)

MANIFIESTO ÁGIL: PRINCIPIOS

9. **La atención continua a la excelencia técnica y al buen diseño mejora la Agilidad**. Cuanta más **calidad** tenga el software en cuanto a diseño y estándares de implementación, más rendimiento se obtiene en las tareas de pruebas, mantenimiento, y mayor reusabilidad.
10. **La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial**. Se estima que el 90% del trabajo realizado no solicitado por el cliente, éste no lo llega a utilizar.
11. **Las mejores arquitecturas, requisitos y diseños emergen de equipos auto-organizados**. (Responsabilidad de todos los miembros del equipo)
12. **A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo para a continuación ajustar y perfeccionar su comportamiento en consecuencia.**

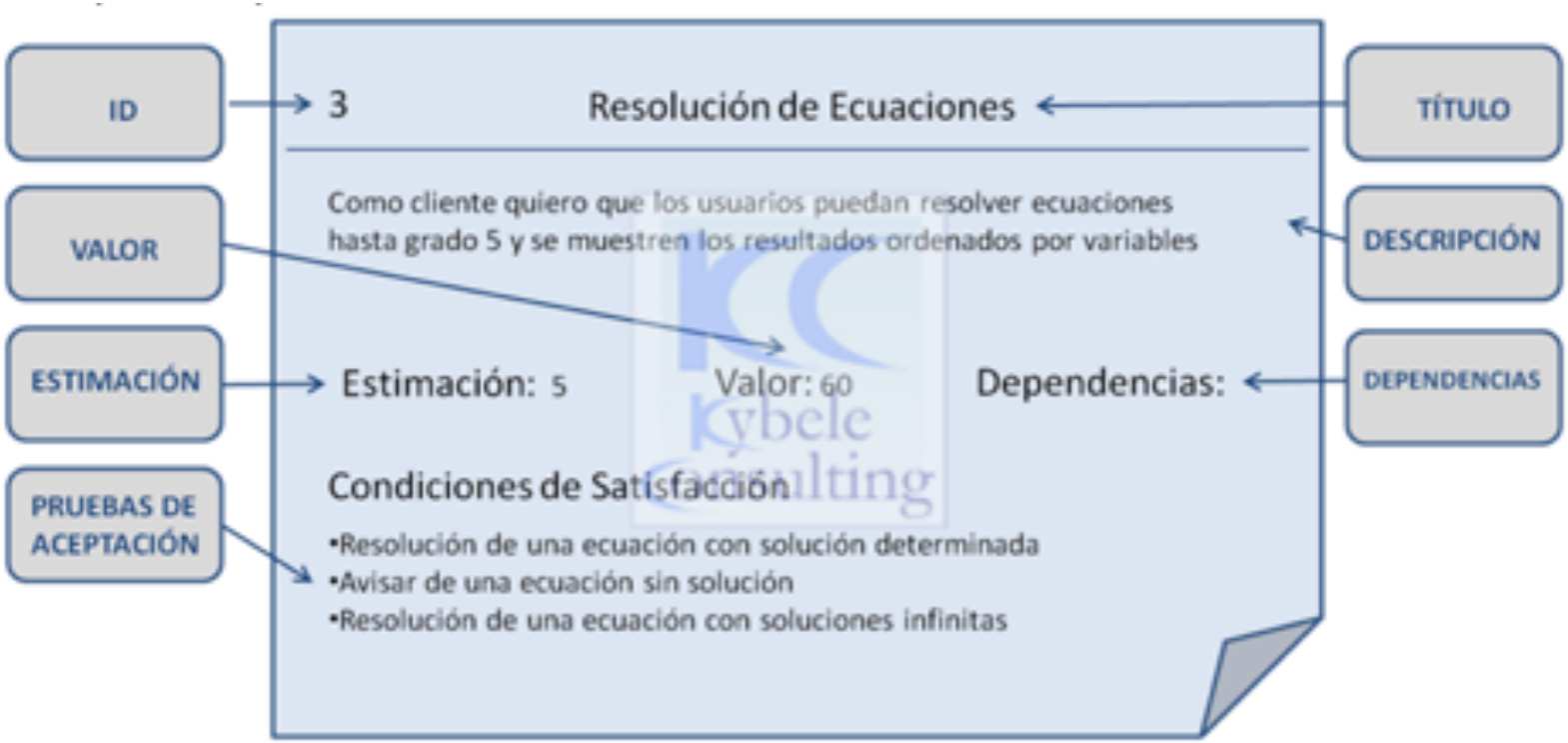
Lectura recomendada: "Metodologías ágiles en el Desarrollo de Software. José H. Canós, Patricio Letelier y Mª Carmen Penadés.

HISTORIAS DE USUARIO

Una **historia de usuario** describe **funcionalidad que será útil para el usuario**, o el comprador de un sistema software. Por definición, las historias de usuario no suelen ni deben tener el nivel de detalle que tiene la especificación de un requisito tradicional.

- Describen brevemente las características que el sistema debe poseer, tanto requisitos funcionales como no funcionales, pero no cómo se desarrollará (**el qué, no el cómo**).
- Es algo que al ser introducido **aporta claro valor al usuario**.
- No son especificación de requisitos.
- Debe ser pequeña, memorizable, y que pudiera ser desarrollada por **un par de programadores en una o un par de semanas**. (No deberían superar el tamaño de una iteración).
- Potente **herramienta de interacción** con el cliente.
- Fomentan la **colaboración, cooperación y conversación** de todos los miembros del equipo.
- Las historias de usuario **se descomponen en tareas** que serán asignadas a los asignadas a los miembros del equipo de desarrollo.

HISTORIAS DE USUARIO



HISTORIAS DE USUARIO

- Descripción: Debe responder a las preguntas:
  - **Quién** se beneficia?
  - **Qué** se quiere?
  - **Cuál** es el beneficio?

Como (rol) quiero (algo) para poder (beneficio)

Las historias de usuario son muy usadas en metodologías ágiles como SCRUM o XP.

HISTORIAS DE USUARIO

- Es importante escribirlas en tarjetas o post-its, para evitar que sean muy grandes.
- Se compone de 3 partes (3Cs):
  - Card: (medio físico)
  - Conversación (discusión)
  - Confirmación (pruebas y verificación)

- Lectura recomendada:

http://www.javiergarzas.com/2011/12/historia-de-usuario-diferente-de-requisito.html

DESARROLLO ÁGIL

El desarrollo ágil de software incluye métodos de ingeniería del software basados en **el desarrollo iterativo e incremental**, donde los requisitos y soluciones evolucionan mediante la colaboración **de grupos auto organizados** y **multidisciplinares**.

INTRODUCCIÓN A SCRUM

- SCRUM es una jugada de rugby que representa la unión de todo el equipo para conseguir un objetivo.
- No es una metodología, es un **marco de trabajo** para la gestión de proyectos ágiles de manera **iterativa** e **incremental**.
- Iteración = **Sprint**

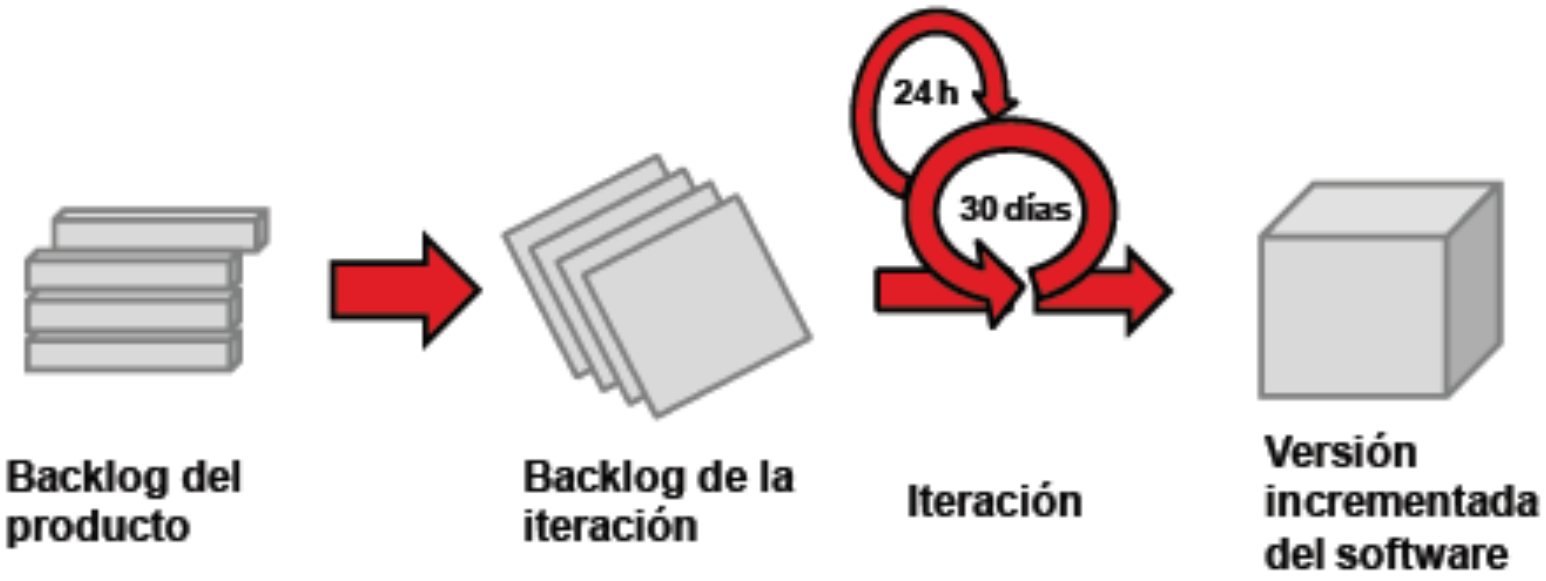
INTRODUCCIÓN A SCRUM

- **OBJETIVOS**:
  - Cómo organizar el ciclo de vida del proyecto ágil de forma iterativa e incremental
  - Cómo conseguir la mejor comunicación e interacción entre los miembros del equipo
- **PILARES**:
  - Ciclo de vida iterativo e incremental
  - Reuniones

SPRINT

- El Sprint es un **periodo de corta duración** (2-4 semanas) en el que **se crea un producto, un prototipo operativo**. Las características que van a implementarse en el Sprint provienen de la **Pila del Producto (Product Backlog**), que contiene una serie de **historias de usuario ordenadas.**

EL FLUJO SCRUM



PILA DE PRODUCTO (PRODUCT BACKLOG)

- La pila de producto es el corazón de Scrum. Es donde empieza todo. La Pila de Producto es, básicamente, **una lista ordenada de historias o funcionalidades**. Cosas que el cliente quiere, descritas usando la terminología del cliente. Llamamos a esto ***historias***, o a veces simplemente ***elementos de la Pila.***

PILA DE PRODUCTO

- Incluyen historias con los siguientes campos:
  - **ID**
  - **Nombre**
  - **Importancia**: o valor que le da el Dueño de Producto.
  - **Estimación inicial:** la valoración inicial del Equipo acerca de cuánto trabajo es necesario para implementar la historia, comparada con otras historias. **Puntos-historia** =“dias-persona” ideales.
  - **Cómo probarlo**: en la demo al final del Sprint. Puede usarse como pseudocodigo del test de aceptación.
  - **Notas**

PUNTO DE HISTORIA

- **Medida de estimación** de las historias de usuario. Mide el volumen de trabajo o complejidad que va a tener cada historia de usuario.
- Un punto historia es una fusión de la cantidad de **esfuerzo** que supone desarrollar la historia de usuario, la **complejidad** de su desarrollo y el **riesgo** inherente.
- No existe una fórmula para calcular los puntos de historia de una historia de usuario.

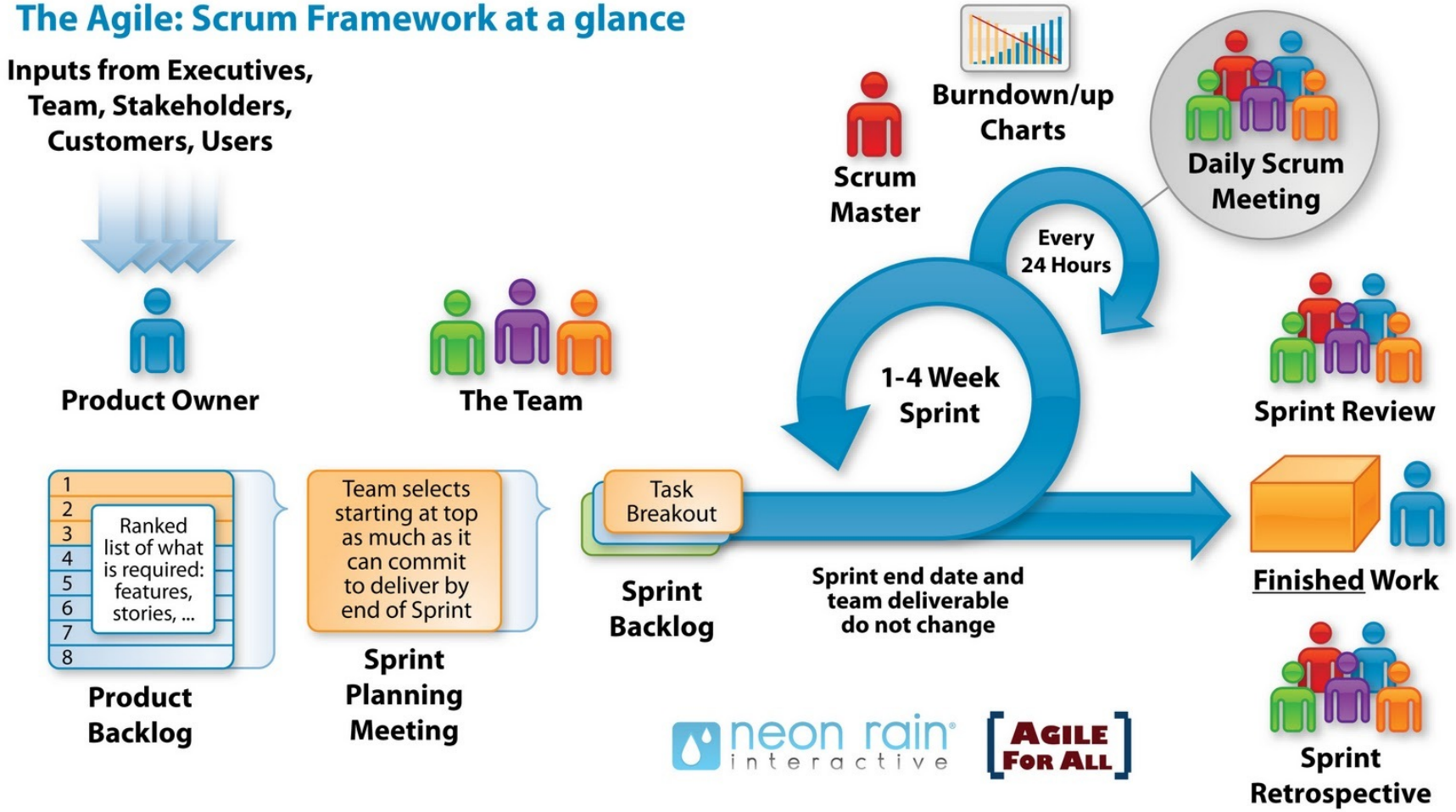
PUNTO DE HISTORIA

- Es responsabilidad del **equipo técnico** del proyecto asignar los puntos de historia a cada historia. Son los que realmente saben lo complejo que va a ser.
- Es una **estimación de alto nivel** que sirve para **comparar y priorizar** historias por el esfuerzo que requiere más el valor que aporta.
- A la hora de asignar puntos de historia a cada historia de usuario, lo que importa son los **valores relativos de una historia frente al resto.**
- Es responsabilidad del Product Owner ordenar y darle prioridad a las historias de usuario.

PILA DEL SPRINT (Sprint Backlog)

- Una vez seleccionadas las historias de usuario que se van a desarrollar en el Sprint, se conforma la **Pila del Sprint (Sprint Backlog)**. Ésta es una lista de **tareas**, donde **se describe cómo el equipo va a implementar las historias de usuario durante el siguiente Sprint**.
- Se mantendrá **inamovible** durante toda la iteración.

The Agile: Scrum Framework at a glance



SCRUM: Roles

Product Owner

- Representa al cliente o usuario final
- Define prioridades para el negocio

Equipo

- Inter-disciplinario
- Auto-organizado

ScrumMaster

- Líder facilitador
- Coach

PRODUCT OWNER

- Figura que representa a los usuarios y recae en **una sola persona**.
- Debe estar **altamente disponible**, tener una **visión clara de lo que** se quiere construir y **ganas de conseguir** que se desarrolle un producto de **éxito**.
- **Responsable de gestionar la Pila del Producto**:
  - Identificar las funcionalidades del producto.
  - Trasladarlas a una lista priorizada.
  - Decidir cuáles deberían estar al principio de la lista en el siguiente Sprint (Es el responsable del retorno de inversión ROI del producto).
  - Repriorizar y refinar continuamente la lista.
- Interactúa de forma activa y regular con el Equipo.
- Revisa los resultados de cada Sprint.

EQUIPO DE DESARROLLO

- Es **multidisciplinar** (engloba toda la experiencia y conocimiento necesarios para desarrollar un producto entregable)

  y auto-organizado (con un amplio margen de autonomía y responsabilidad).

- **Responsabilidad**:
  - Desarrollar el producto
  - Proporcionar ideas al Dueño del Producto

EQUIPO DE DESARROLLO

- Suele tener 7 ±2 personas.
- Son más productivos y efectivos si :
  - están dedicados al 100% en un producto durante el Sprint.
  - Son estables (prácticamente las mismas personas)
- Su objetivo es **conseguir terminar la funcionalidad comprometida** en el Sprint. Para ello podrán emprender tareas con las que se sienten menos familiarizados con el objetivo de ayudar a completar un elemento. Esto implica un **aprendizaje múltiple**: cada persona tendrá ciertas fortalezas, pero también continuará aprendiendo otras especialidades.

SCRUM MASTER

- **Es coach y formador**: forma, entrena y guía al resto de participantes en el **uso correcto de Scrum**.
- Sirve al Equipo **facilitándole el proceso** mientras que este se organiza y se gestiona por su cuenta:
  - Le ayuda a eliminar impedimentos
  - Le protege de interferencias externas
  - Le ayuda a adoptar prácticas de desarrollo modernas
- **No debe** ser la misma persona que el **Dueño de Producto**, ya que el enfoque es muy diferente.

STAKEHOLDERS

- Otros participantes del proyecto que contribuyen al éxito del producto, como por ejemplo los gerentes, clientes, usuarios finales, etc.

~~JEFE DE PROYECTO~~

- En Scrum **NO EXISTE** el rol de jefe de proyecto.
- Las responsabilidades tradicionales del jefe de proyecto han sido divididas y reasignadas entre los tres roles de Scrum, **sobre todo entre el Equipo y el Dueño de Producto** más que en el ScrumMaster.

DEFINICIÓN DE “DONE” (TERMINADO)

- El resultado de cada Sprint se denomina oficialmente “**Incremento de Producto Potencialmente Entregable**”
- Antes del primer Sprint, el Dueño de Producto y el Equipo deben acordar una Definición de “Done”, que será un subconjunto de las actividades necesarias para crear un Incremento de Producto Potencialmente Entregable. El Equipo planificará su Sprint atendiendo a esta Definición de “Terminado”.
- Un buen Dueño de Producto siempre querrá que la Definición de “Terminado” sea lo más cercana posible a Potencialmente Entregable, ya que esto incrementará la transparencia en el desarrollo y reducirá el *retraso* y el *riesgo*. El trabajo retrasado se denomina en **ocasiones *trabajo sin terminar***.
- Un Equipo Scrum mejora de forma continua, lo que se refleja en la extensión de su Definición de “Terminado”.