# Metodologías Agiles



*Inst: Pablo Ortiz*

# SCRUM

Metodología Ágil de Desarrollo de Software

*Inst: Pablo Ortiz*

# METODOLOGÍAS ÁGILES DE DESARROLLO

El término ágil surge como iniciativa de un conjunto de expertos en el área de desarrollo de software con el fin de optimizar el proceso de creación del mismo, el cual era caracterizado por ser rígido y con mucha documentación. El punto de partida fue el manifiesto ágil, el cual es un documento donde se detalla todo lo que involucra la filosofía “ágil”. 2001

# MANIFIESTO ÁGIL

1. Al individuo y las interacciones del equipo de desarrollo sobre el proceso y las herramientas.

2. Desarrollar software que funcione más que la documentación del mismo.

3. La colaboración con el cliente más que la negociación de su contrato.

4. Responde a los cambios más que seguir con el plan establecido.

# PRINCIPIOS DE CALIDAD

1. La prioridad más alta es satisfacer al cliente a través de la entrega pronta y continua de software valioso.

2. Son bienvenidos los requerimientos cambiantes, aun en una etapa avanzada del desarrollo. Los procesos ágiles dominan el cambio para provecho de la ventaja competitiva del cliente.

3. Entregar con frecuencia software que funcione, de dos semanas a un par de meses, de preferencia lo más pronto que se pueda.

4. Las personas de negocios y los desarrolladores deben trabajar juntos, a diario y durante todo el proyecto.

5. Hay que desarrollar los proyectos con individuos motivados. Debe darse a éstos el ambiente y el apoyo que necesiten, y confiar en que harán el trabajo.

# PRINCIPIOS DE CALIDAD

6. El método más eficiente y eficaz para transmitir información a los integrantes de un equipo de desarrollo, y entre éstos, es la conversación cara a cara.

7. La medida principal de avance es el software que funciona.

8. Los procesos ágiles promueven el desarrollo sostenible. Los patrocinadores, desarrolladores y usuarios deben poder mantener un ritmo constante en forma indefinida.

9. La atención continua a la excelencia técnica y el buen diseño mejora la agilidad.

10. Es esencial la simplicidad: el arte de maximizar la cantidad de trabajo no realizado.

11. Las mejores arquitecturas, requerimientos y diseños surgen de los equipos con organización propia.

12. El equipo reflexiona a intervalos regulares sobre cómo ser más eficaz, para después afinar y ajustar su comportamiento en consecuencia

# VENTAJAS

- **Mejoran la satisfacción del cliente** dado que se involucrará y comprometerá a lo largo de todo el proyecto. En cada etapa se informará al cliente de los logros y progresos del mismo, con la visión de involucrarlo directamente para sumar su experiencia y conocimiento, y así, optimizar las características del producto final obteniendo en todo momento una visión completa de su estado.

- **Mejora de la motivación e implicación del equipo de desarrollo**. las metodologías ágiles permiten a todos los miembros del equipo conocer el estado del proyecto en cualquier momento, así, los compromisos son negociados y aceptados por todos los miembros del equipo.

- **Permite ahorrar tiempo y costes.** El desarrollo ágil trabaja de un modo más eficiente y rápido, y con ello, se cumple de forma estricta el presupuesto y los plazos pactados dentro de un proyecto.

- **Mayor velocidad y eficiencia**. Una de las máximas de su aplicación es que se trabaja a través de entregas parciales del producto, de este modo, es posible entregar en el menor intervalo de tiempo posible una versión mucho más funcional del producto.

# VENTAJAS

- **Eliminar cualquier característica innecesaria del producto**. Gracias a las entregas parciales (centradas en entregar en primer lugar aquellas funcionalidades que aportan valor) y a la implicación del cliente.

- **Mejorar la calidad del producto**. La continua interacción entre los desarrolladores y los clientes tiene como objetivo asegurar que el producto final sea exactamente lo que el cliente busca y necesita.

- **Alertar de forma rápida tanto de errores o problemas** que puedan sucederse a lo largo del proyecto. Es posible dar respuesta a todos aquellos problemas que puedan darse desde el inicio, con lo que mejoramos en costes y entrega.

- **Permiten rentabilizar nuestras inversiones,** y es que, gracias a la realización de entregas tempranas el cliente tendrá rápido acceso a aquellas funcionalidades que aportan valor acelerando el retorno de la inversión.


# SCRUM

*Inst: Pablo Ortiz*

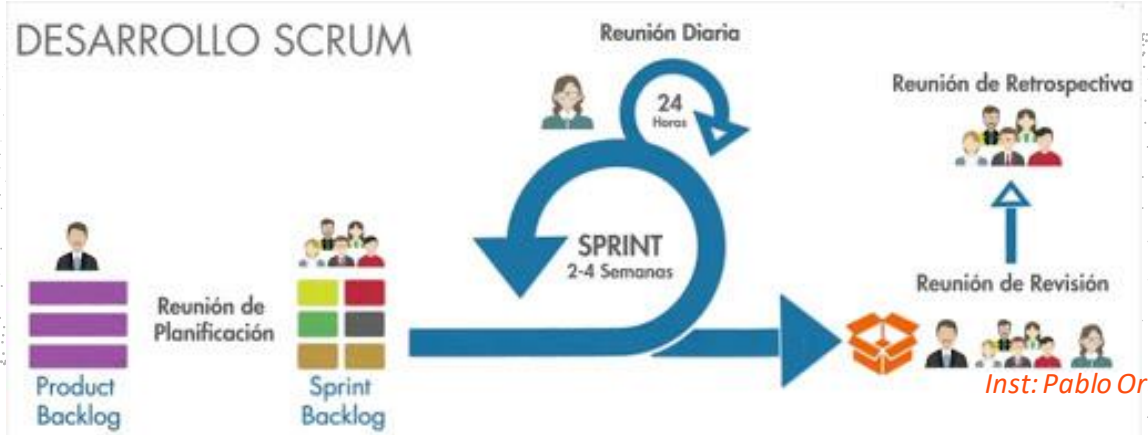
# METOLOGIA SRUM

Scrum es un marco de trabajo para desarrollo ágil de software. Es un proceso en el que se aplican de manera regular un conjunto de buenas prácticas para trabajar colaborativamente, en equipo y obtener el mejor resultado posible del proyecto.

En la etapa de desarrollo encontramos lo que se conoce como interacciones del proceso o Sprint, es decir, entregas regulares y parciales del producto final.

Existen tres pilares fundamentales que soportan el control del proceso empírico los cuales son:

- Transparencia
- Inspección
- Adaptación



# VALORES


**CORAGE**

*El equipo scrum tiene el coraje de hacer lo correcto y trabajar en problemas difíciles*

**ATENCIÓN**

*Todos se centran en el trabajo del sprint y los objetivos del equipo scrum*

**COMPROMISO**

*Las personas se comprometen personalmente a lograr los objetivos del equipo scrum*

**EL RESPETO**

*Los miembros del equipo scrum se respetan mutuamente para ser personas capaces e independientes.*

**FRANQUEZA**

*El equipo scrum y sus partes interesadas acuerdan ser abiertos sobre todo el trabajo y los desafíos para realizar el trabajo.*

*Inst: Pablo Ortiz*


# ROLES SCRUM

*Inst: Pablo Ortiz*

# ROLES SCRUM

***Product Owner,*** *E*s la persona responsable de transmitir al equipo de desarrollo la visión del producto que se desea crear, aportando la perspectiva de negocio.

- **Expresar** claramente los requerimientos.
- **Ordenar** los elementos de la lista de requerimientos para alcanzar los objetivos de la mejor manera posible.
- **Optimizar** el valor del trabajo desempeñado por el equipo de desarrollo.
- **Asegurar** que la lista de requerimientos es visible, transparente y claro para todos.
- **Asegurar** que el equipo de desarrollo entiende los elementos de la lista de requerimientos.

# Roles de Scrum

By Javier Garzás, Ana María García (2014)

## Responsabilidades

- Decidir qué construir... ¡y que no!
- Recoger y tener claros los requisitos del software.
- Definir buenas historias de usuario.
- Fijar criterios de aceptacion para cada historia de usuario.
- Ordenar y priorizar los items del Product Backlog.
- Definir el producto minimo viable.
- Acordar junto al resto del equipo una definicion de DONE.
- Definir el plan de releases.
- Validar entregas (Sprint Review)

## Product Owner

Debe participar en las reuniones

- Daily meeting (Participacion
- Sprint Planning
- Sprint Review
- Sprint Retrospective

El Product Owner no debe dar ordenes al equipo.

El equipo no debe trabajar en otros requisitos distintos a los que el Product Owner incluya en el Backlog.

- Estar disponible y accesible para el equipo.
- Es el responsable de cancelar el sprint si ocurre un imprevisto extremo.
- Asegurarse de que el Product Backlog es visible para todo el mundo.
- Asegurarse de que todo el mundo entiende los items del Product Backlog.

**El Product Owner debe conocer la velocidad del equipo, para realizar estimaciones de cuando estaran implementadas las necesidades en el producto.**



Inst: Pablo Ortiz

# ROLES SCRUM


***Scrum Master,*** es un líder que está bajo el servicio del equipo scrum, este miembro ayuda al equipo y a los clientes externos a comprender las Interacciones que pueden ser de ayuda y cuáles no lo son, además él es el encargado de asegurar que el equipo adopte las teorías, prácticas y reglas de la metodología scrum.

- **Facilitar** reuniones y eventos del equipo
- **Coaching** a los miembros del equipo, mediando a través de conflictos, ayudando a las decisiones y al equipo a ser auto-organizado.
- **Aprendizaje** continuo, ayudando al equipo a crear radiadores de información.
- **Remover** impedimentos.
- **Reflejar** los valores ágiles y de Scrum, recordando al equipo sus acuerdos, ayudando al equipo continuamente a mejorar sus procesos y preguntar cuestiones abiertas.

# Roles de Scrum

*By Javier Garzás, Ana María García (2014)*

## Responsabilidades

- Planifica la implantación de Scrum junto con la organización.
- Ayuda a la organización a entender qué interacciones con el equipo aportan valor y cuáles no.
- Ayuda al Product Owner a entender la agilidad.
- Ayuda al Product Owner a maximizar el valor de negocio.
- Enseña al Product Owner a priorizar y gestionar efectivamente el Product Backlog.
- Ayuda al equipo de desarrollo a convertirse en auto-organizado y multifuncional.
- Soluciona posibles impedimentos que pudieran surgir durante el Sprint.
- Se asegura de que haya una definición de DONE



## Scrum Master

Debe participar en las reuniones y asegurarse de que cumplan el tiempo y el objetivo establecido

- Daily meeting
- Sprint Planning
- Sprint Review
- Sprint Retrospective

El Scrum Master no es lo mismo que el Product Owner. El Product Owner tiene una visión más de negocio, mientras que el Scrum Master se encarga de que todo el equipo entienda Scrum y lo aplique correctamente.

- Ayuda a que las posibles mejoras detectadas en la retrospectiva del Sprint se lleven a cabo.
- Junto con el equipo de desarrollo, actualiza el trabajo en progreso (burndown chart).
- Se asegura y promueve buenas prácticas de programación.
- Realiza cursos para aprender Scrum si es necesario.

El Scrum Master es el responsable de asegurar que se sigue Scrum.

*Inst: Pablo Ortiz*

# ROLES SCRUM

***Equipo de desarrollo,*** equipo responsable de desarrollar y entregar el producto. Mantiene una organización horizontal en la que cada miembro del equipo se auto-gestiona y organiza libremente en la definición y ejecución de los distintos sprints.

- Son **auto-organizados.**
- Los equipos de desarrollo son **multifuncionales.**
- Scrum no reconoce **títulos** para los miembros de un equipo de desarrollo.
- La responsabilidad del cumplimiento recae en todo el **equipo.**

# Roles de Scrum

*By Javier Garzás, Ana María García (2014)*

## Responsabilidades

Trabajar para cumplir el Sprint Goal.

Saber buenas practicas de programacion (TDD, integracion continua etc).

Implementar pruebas unitarias y de aceptacion.

Estar motivados.

Trabajo de calidad y mejora continua (por ejemplo, refactorizaciones)

Actualizar el trabajo en progreso.

javiergarzas.com 233 grados de TI

## El equipo de desarrollo

### Debe ser

- De 3-7 personas
- Auto-organizado
- Multifuncional

### Asiste a

**Daily meeting**

Encuentran impedimentos y se los comunican al Scrum Master.

**Sprint Planning**

Junto con el Scrum Master, planifican los items que van a implementarse en el sprint.

Estiman las historias de usuario y tareas.

**Sprint Review**

Hacer la demo.

**Sprint Retrospective**

Ven como mejorar para el siguiente sprint.

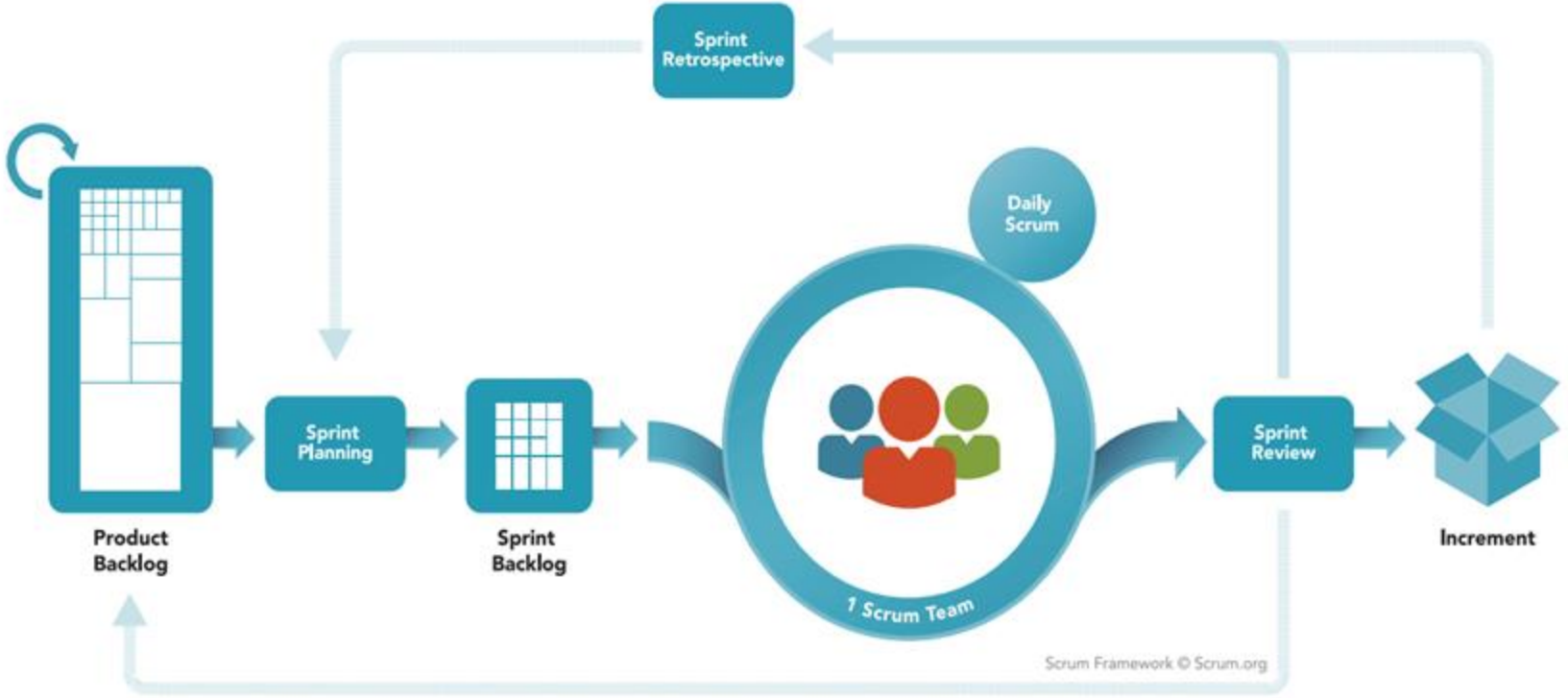
*Inst: Pablo Ortiz*

# ROLES SCRUM

***Stakeholders,*** *C*onjunto de personas que no forman parte directa del proceso de desarrollo pero que si deben ser tenidos en cuenta, por ser personas interesadas en el mismo, tales como directores, gerentes, comerciales etc.

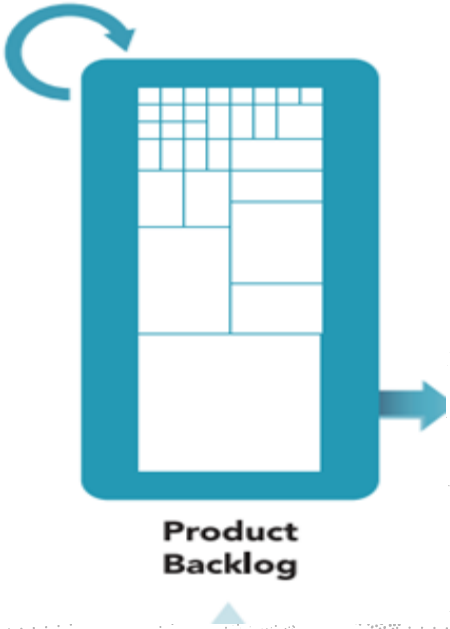
***Usuarios,*** *A*l igual que los Stakeholders no forman parte del proceso de creación directamente (podrían estar en la fase de revisión de entregables si se considera necesario). Son los destinatarios finales de la aplicación a desarrollar, el público objetivo del mismo

# CICLO SCRUM




*Inst: Pablo Ortiz*

# PRODUCT BACKLOG




**La Lista de Producto:** es una lista ordenada de todo lo que se conoce que es necesario en el producto. Es la única fuente de requisitos para cualquier cambio a realizarse en el producto.

El Dueño de Producto (Product Owner) es el responsable de la Lista de Producto, incluyendo su contenido, disponibilidad y ordenación.

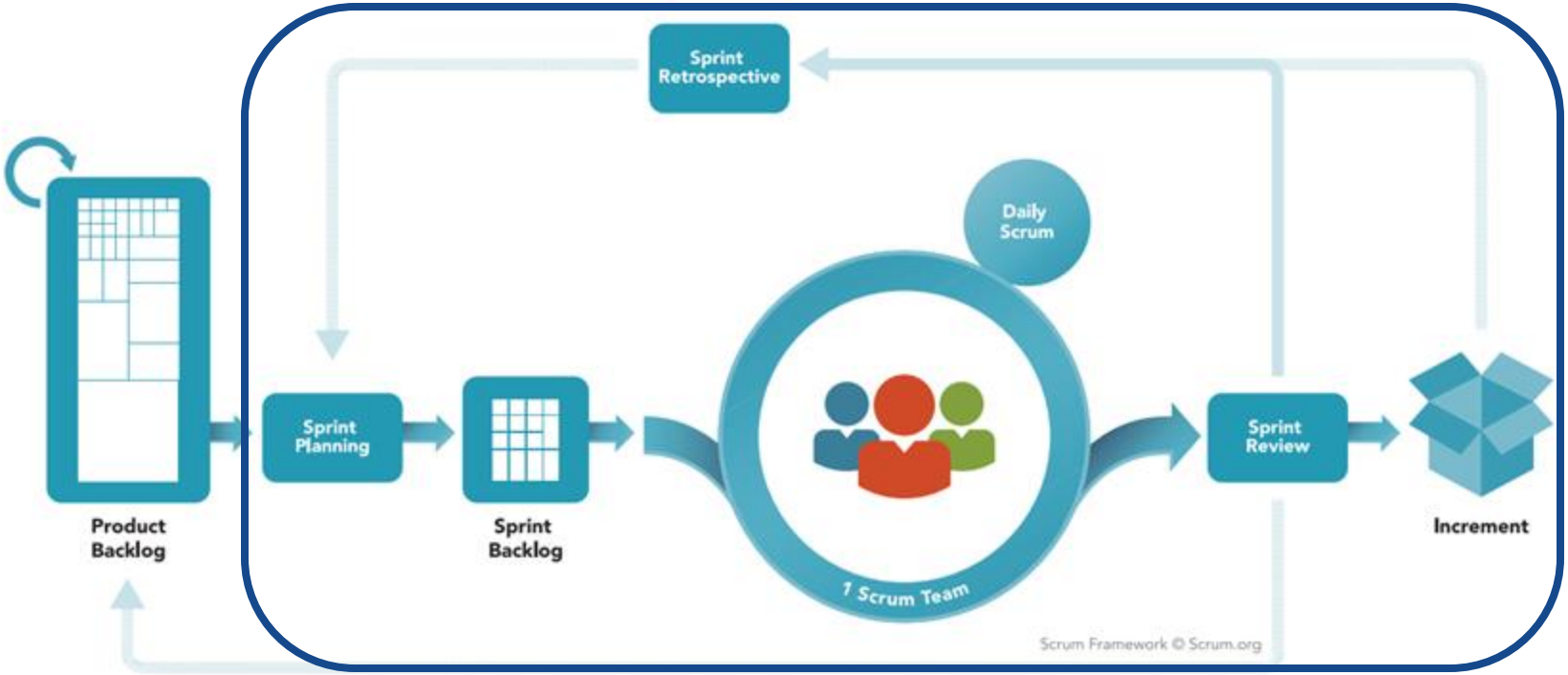
Una Lista de Producto nunca está completa..

## SPRINT

El **Sprint** corresponde al bloque de tiempo (entre 1 a 4 semanas) durante el cual se crea un incremento de producto "*Terminado*", utilizable y potencialmente desplegable. Cada nuevo Sprint comienza inmediatamente después de la finalización del Sprint previo.

Preguntas:
- Que hiciste ayer.
- Que vas a hacer mañana
- que Problemas se encontraron

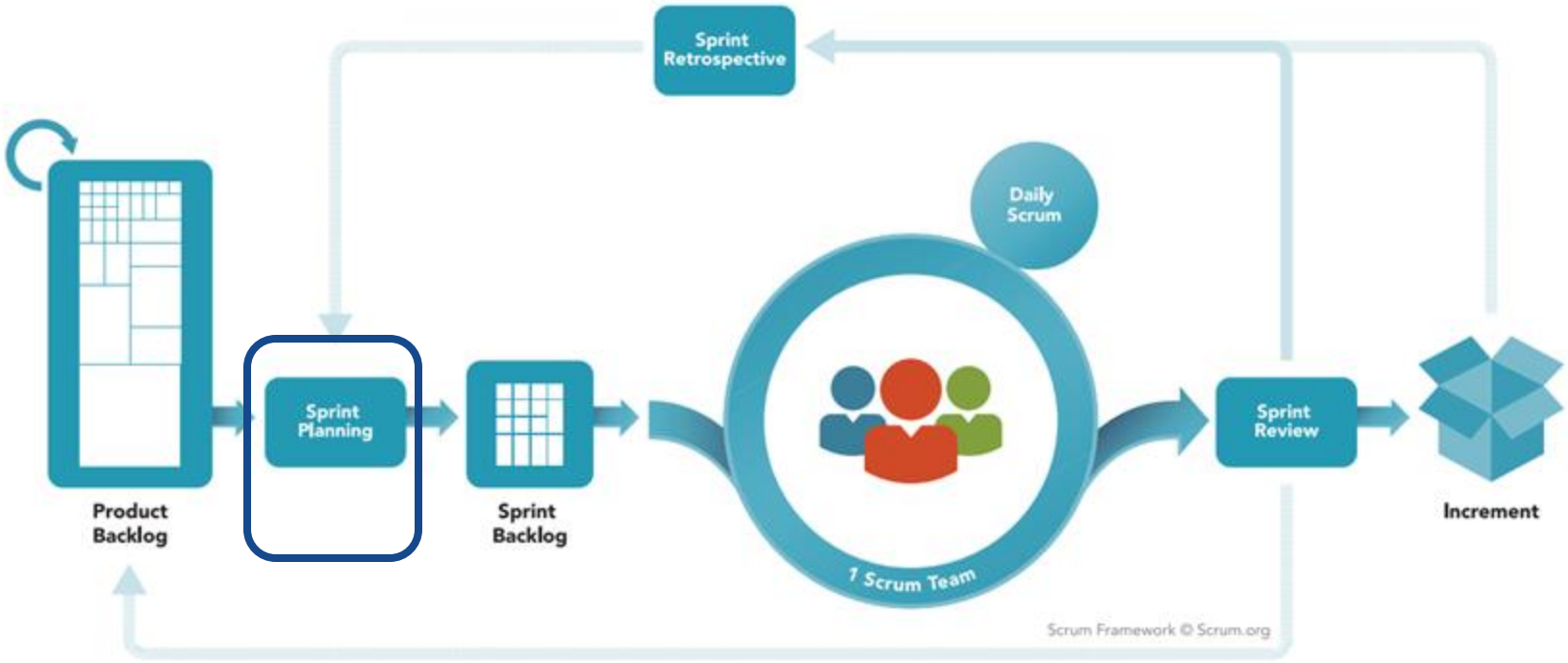
## SPRINT


Sprint Retrospective
Daily Scrum
Product Backlog
Sprint Planning
Sprint Backlog
1 Scrum Team
Sprint Review
Increment
Scrum Framework © Scrum.org

## SPRINT PLANING

El objetivo buscado en el **Sprint Planning** es identificar "*qué*" es lo que el equipo de desarrollo va a realizar durante el Sprint, y "*cómo*", es decir, todos aquellos PBLs que el equipo se comprometerá a transformar en un incremento funcional potencialmente entregable.

## SPRINT PLANING

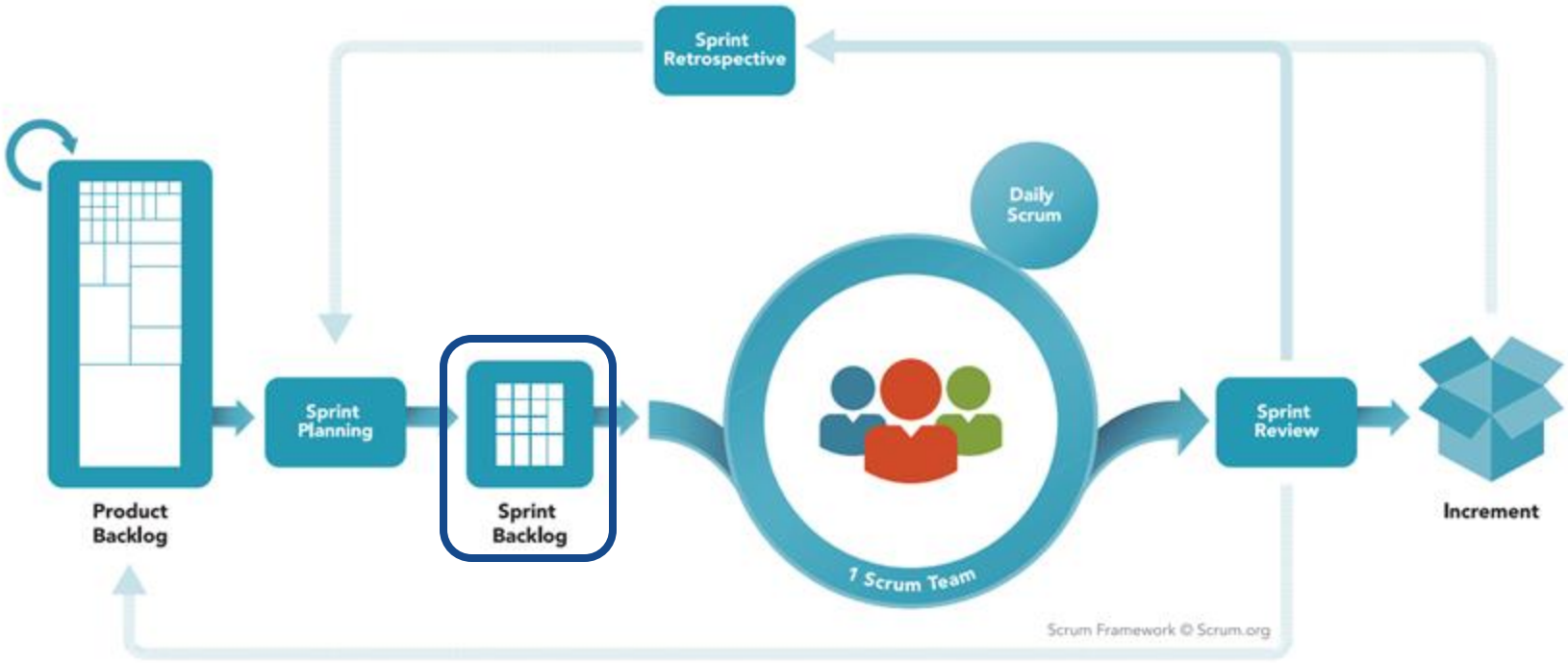


# SPRINT BACKLOG

El **Sprint Backlog** o también conocido como **Lista de Pendientes del Sprint** es el conjunto de elementos del Product Backlog seleccionados para el Sprint.

Es una predicción hecha por el Equipo de Desarrollo acerca de qué funcionalidad formará parte del próximo Incremento y del trabajo necesario para entregar esa funcionalidad en un Incremento “Terminado”.

# SPRINT BACKLOG



## DAILY MEETING

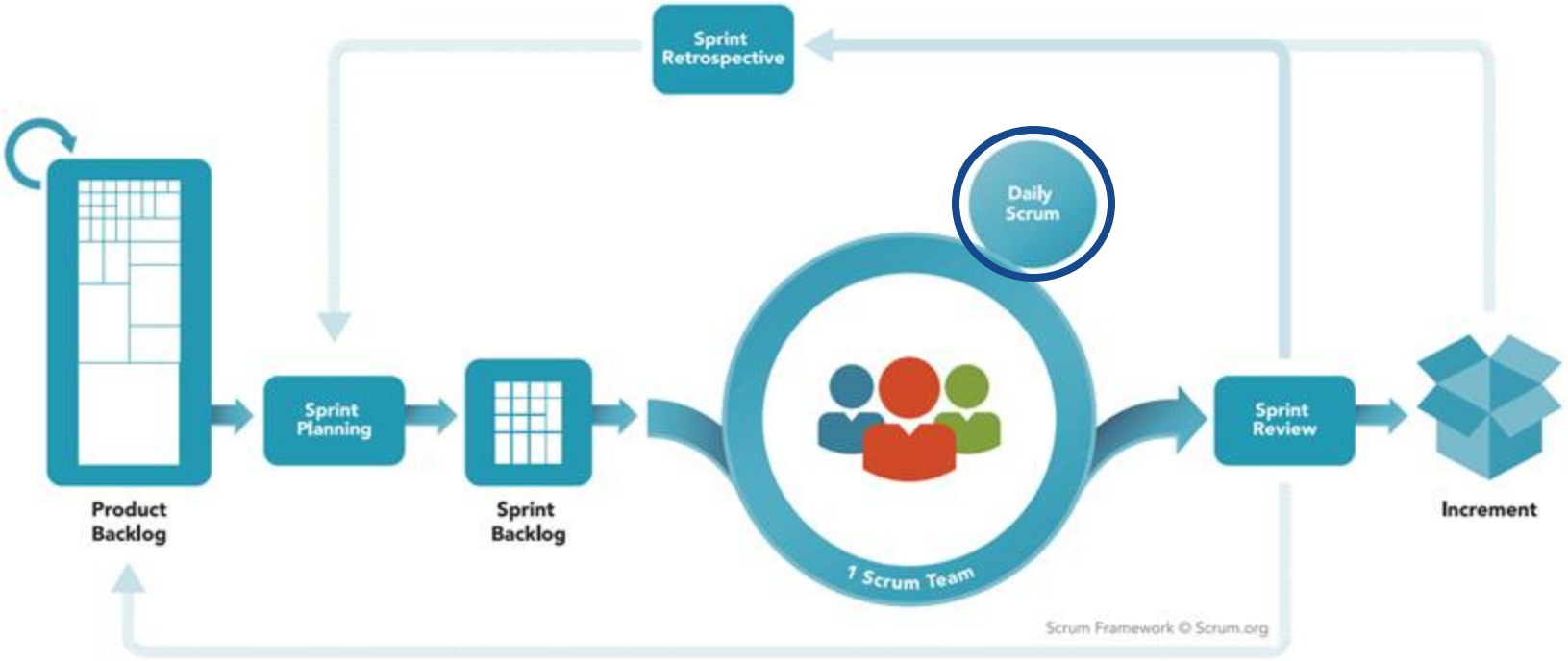

Es una reunión diaria en la que los asistentes suelen participar mientras están de pie. La incomodidad de estar de pie por largos períodos tiene la intención de mantener las reuniones cortas.

Los objetivos de la **Daily Meeting** son:

- Incremento de la comunicación dentro del equipo de proyecto,
- aumentar y explicitar los compromisos asumidos entre los miembros del equipo de desarrollo y
- dar visibilidad a los impedimentos que surjan del trabajo que está siendo realizando y que muchas veces nos impiden lograr los objetivos.

*Inst: Pablo Ortiz*

## DAILY MEETING



# DAILY MEETING

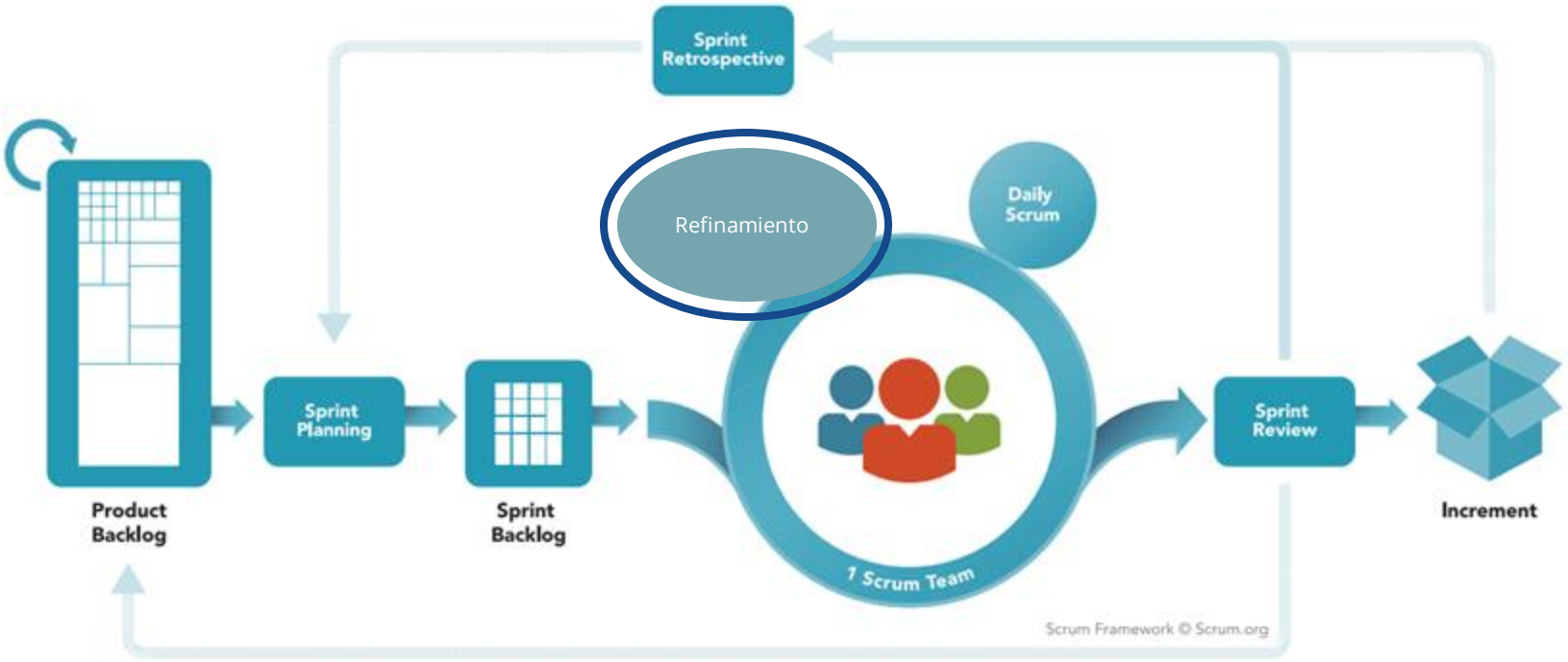

*Inst: Pablo Ortiz*

# REFINAMIENTO

El **Refinamiento** no corresponde a un evento de Scrum, pero si se reconoce como una buena práctica.

Esta actividad en Scrum tiene como objetivo presentar al equipo los ítems del Product Backlog que tienen una probabilidad alta de ser tomados en los próximos Sprints (tanto para el siguiente sprint como para los 2 o 3 próximos) y discutir aspectos sobre ellos.

# REFINAMIENTO

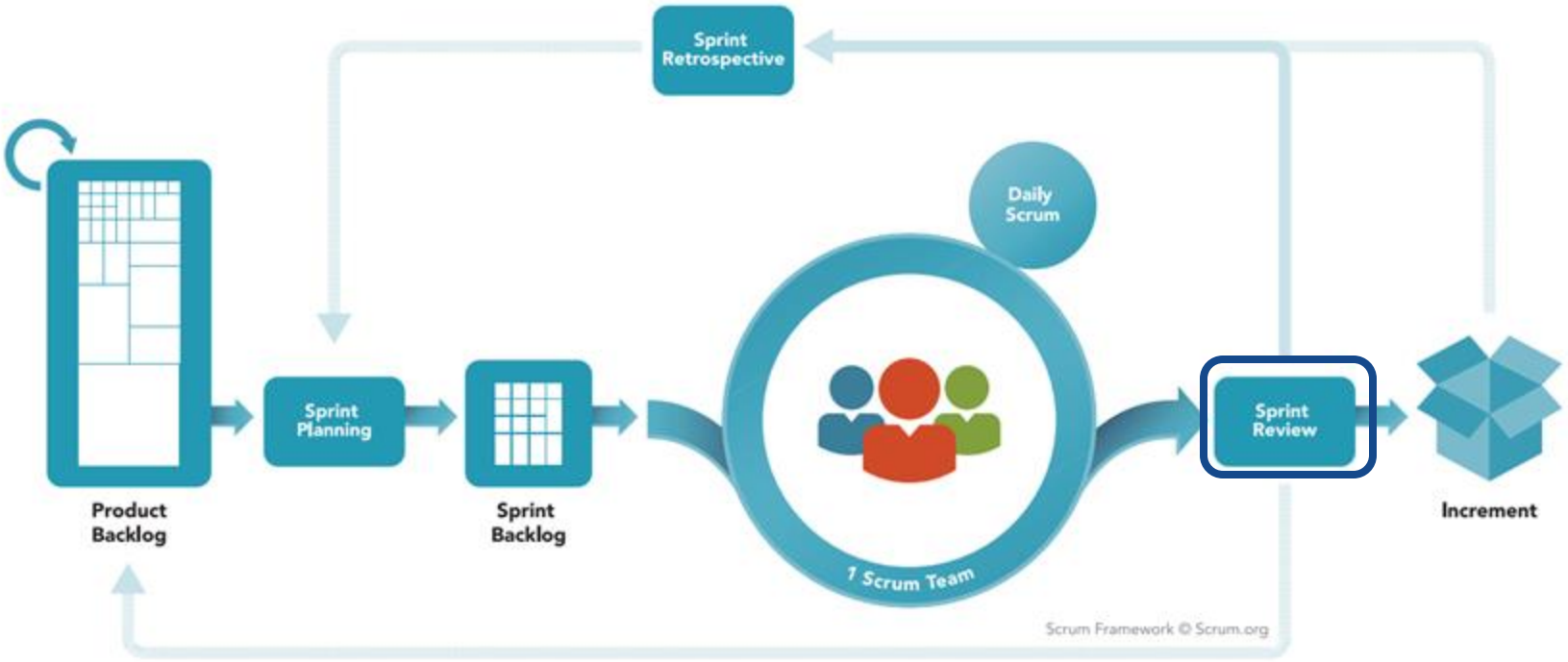


## SPRINT REVIEW

En la **Sprint Review** se revisa el resultado del Sprint.

Cuando decimos “resultado” hablamos de “producto utilizable” y “potencialmente entregable” que él o los interesados utilizan y evalúan durante esta misma reunión, aceptando o rechazando así las funcionalidades construidas.

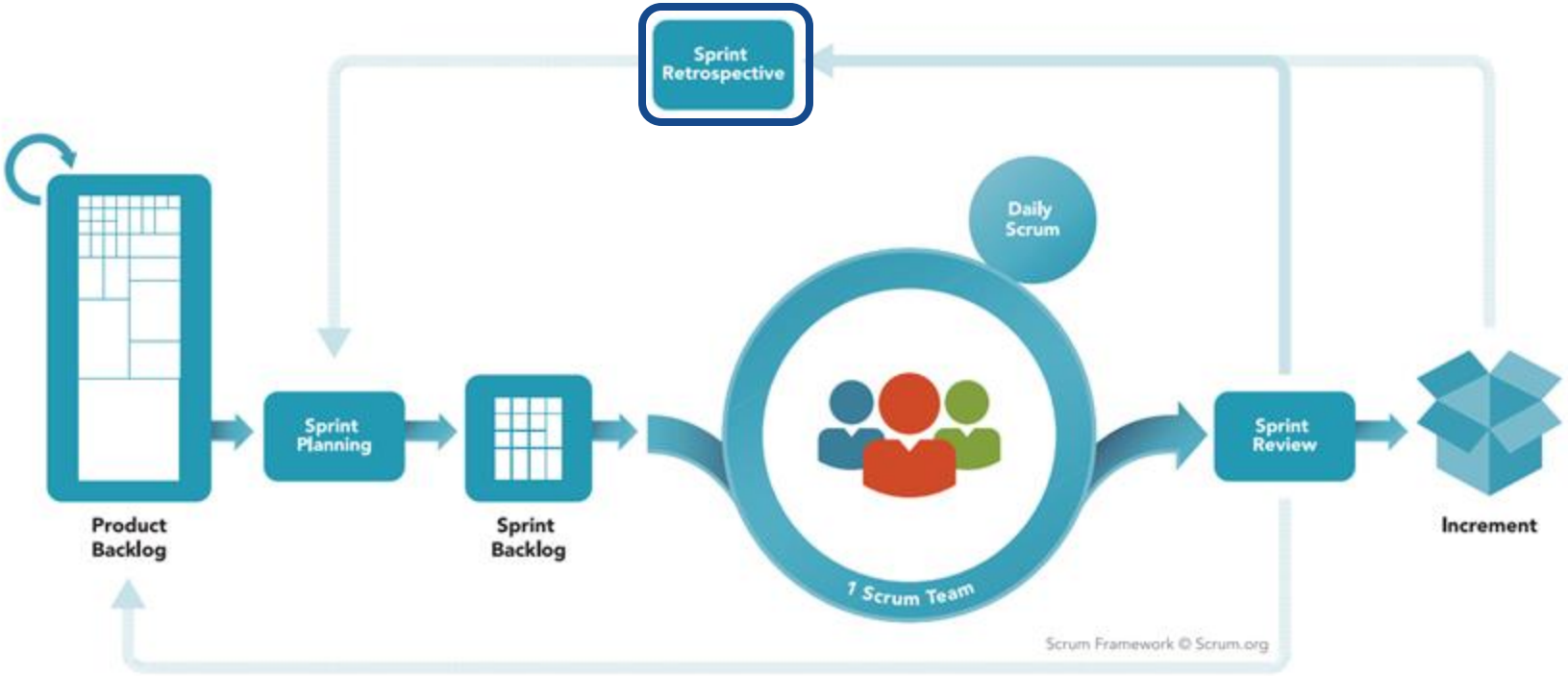
## SPRINT REVIEW



## SPRINT RETROSPEECTIVE

La **Sprint Restrospective** o **Retrospectiva del Sprint** del equipo es el corazón de la mejora continua y las prácticas emergentes. Mediante el mecanismo de introspección, el equipo reflexiona sobre la forma en la que realizó su trabajo y los acontecimientos que sucedieron en el Sprint que acaba de concluir para mejorar sus prácticas.

## SPRINT RETROSPEECTIVE



# Historias de usuario

## ¿Por qué?

Utilizamos las historias de usuario porque siguen los principios básicos de requerimientos ágiles:

- Potencian la participación del equipo en la toma de decisiones.
- Se crean y evolucionan a medida que el proyecto avanza.
- Son peticiones concretas y pequeñas.
- Contiene la información imprescindible. Menos es más.
- Apoyan la cooperación, colaboración y conversación entre los miembros del equipo, lo que es fundamental.

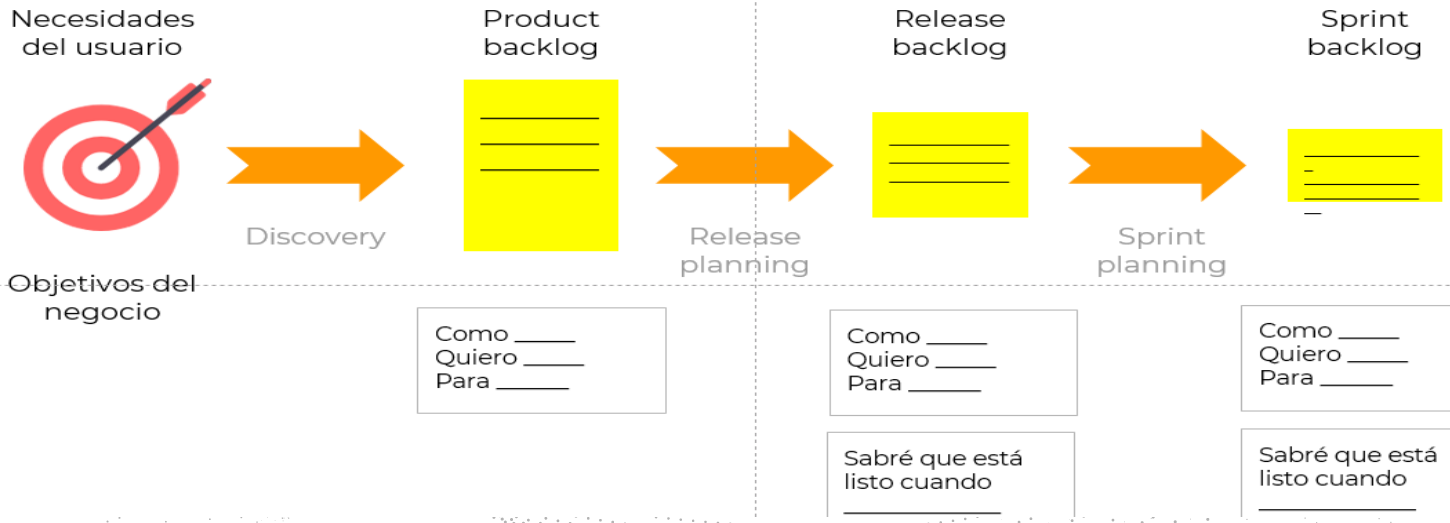
## Que es?

En agilidad las **historias de usuario** son el reemplazo escrito de los **requerimientos de usuario,** se escriben en el **lenguaje propio de los usuarios** y describen **que** debería "**construir**" y "**entregar**" el equipo de desarrollo.

Son una invitación a una **conversación** y no una descripción extensiva.

## HISTORIA DE USUARIO

Una historia de usuario es una representación de un requisito escrito en una o dos frases utilizando el lenguaje común del usuario.



# HISTORIA DE USUARIO
# EJEMPLO

Historia: Registrarse

Como: Lector del Blog

Quiero: suscribirme al Blog

Para: poder realizar comentarios a las entradas de mi interés

2

Historia: Ingresar al Blog

Como: Lector del Blog

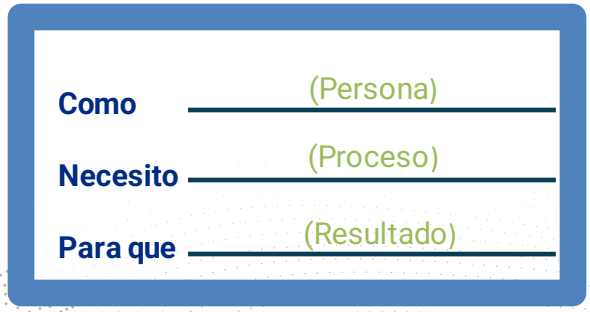
Quiero: Ingresar al Blog con mi usuario y contraseña

Para: realizar comentarios y mantenerme en contacto con otros lectores que compartan mis intereses

2

# HISTORIAS DE USUARIO

Ejemplo:
Como padre, necesito un colegio para que mis hijos puedan estudiar




Yo como padre, necesito un colegio para que mis hijos puedan estudiar

**Criterio de Aceptación**

Edificio infantil
Piscina
Cancha de Futbol
Edificio principal
Casino

*Inst: Pablo Ortiz*

# HISTORIAS DE USUARIO

**Las 3 C: Card, Conversation, Confirmation**

1. **Tarjeta**: es la descripción escrita de la historia, que sirve como identificación, recordatorio y también ayuda a planificar.
2. **Conversación**: es el núcleo de la historia; es el diálogo que ocurre con los usuarios, notas, grabaciones, prototipos y documentos.
3. **Confirmación**: el criterio de las pruebas de aceptación que el usuario va a utilizar para confirmar que la historia fue terminada.

Card

Conversation

Confirmation

*Inst: Pablo Ortiz*

# HISTORIAS DE USUARIO

Una buena historia de usuario también sigue el modelo de INVEST: Independiente, Negociable, Estimable, Pequeña (Small), y Testeable.

**I** Independent

**N** Negotiable

**V** Valuable

**E** Estimable

**S** Small sized

**T** Testable

*Inst: Pablo Ortiz*

# HISTORIAS DE USUARIO

- **Independiente** - una historia debería ser independiente de otras (lo más posible). Las dependencias entre las historias hace que sea más difícil planificar, priorizar y estimar. A menudo, se puede reducir las dependencias haciendo una combinación de historias, o partiendo historias de forma diferente.
- **Negociable** - una historia de usuario es negociable. La "tarjeta" de la historia es tan sólo una descripción corta que no incluye detalles. Los detalles se trabajan durante la etapa de "Conversación". Una tarjeta con demasiados detalles limita la conversación con el cliente.
- **Valiosa** - cada historia tiene que tener valor para el cliente (para el usuario o para el comprador). Una forma muy buena de generar historias valiosas es hacer que el cliente las escriba. Una vez que el cliente se de cuenta que la historia no es un contrato y es negociable, van a sentirse mucho más cómodos para escribir historias.
- **Estimable** - los desarrolladores necesitan poder estimar una historia de usuario para permitir que se pueda priorizar y planificar la historia. Los problemas que pueden impedirle a los desarrolladores estimar una historia son: falta de conocimiento del dominio (en cuyo caso se necesita más Negociación / Conversación); o si la historia es muy grande (en cuyo caso se necesita descomponer la historia en historias más pequeñas).
- **Pequeña** - una buena historia debe ser pequeña en esfuerzo, generalmente representando no más de 2-3 personas/semana de trabajo. Una historia que es más grande va a tener más errores asociados a la estimación y alcance.
- **Testeable** - una historia necesita poder probarse para que ocurra la etapa de "Confirmación". Recordemos que desarrollamos aquello que no podemos probar. Si no podemos probarlo, nunca vamos a saber si lo terminamos. Un ejemplo de historia no testeable sería: "el software tiene que ser fácil de usar".

*Inst: Pablo Ortiz*

Criterio de Aceptación

Son **enunciados** que se describen desde el **punto de vista del usuario** cuándo una historia se puede considerar **hecha** y **correctamente** implementada.

Trae 6 huevos y si hay papa, trae 9

+

o

# HISTORIAS DE USUARIO


*Inst: Pablo Ortiz*