udp Escuela de Informática y Telecomunicaciones
FACULTAD DE INGENIERÍA Y CIENCIAS

# Tutoría para Examen de Grado

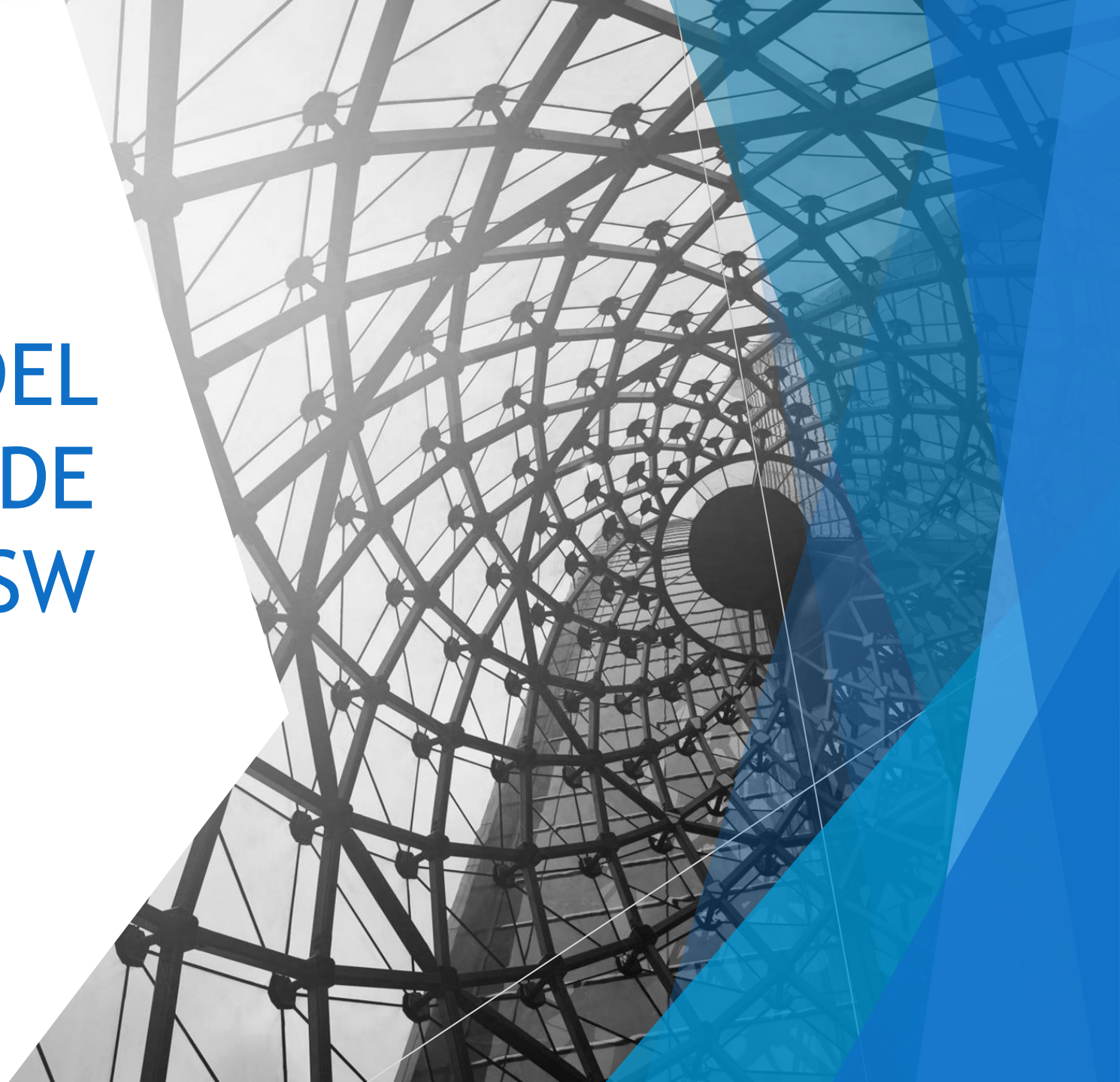
Sesión 4: Etapas, Modelos y Metodologías de Desarrollo de Software

*Docente: Mauricio Hidalgo Barrientos*

# Temario de la sesión

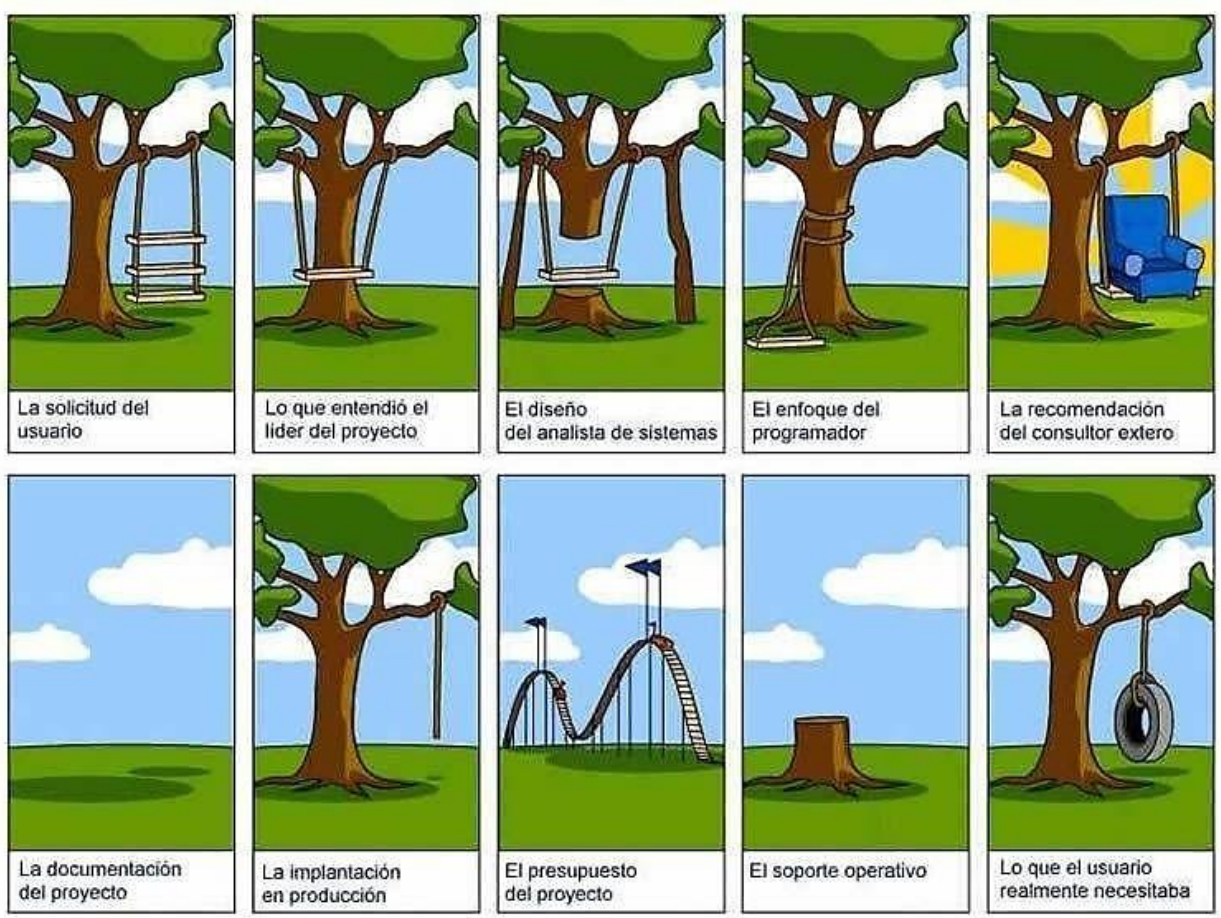
- Recordar las 7 etapas del proceso de Desarrollo de Software
- Recordar Modelos de Desarrollo de Software
- Recordar Metodologías Ágiles de Desarrollo de Software

# LAS 7 ETAPAS DEL PROCESO DE DESARROLLO DE SW



# Introducción

- Los proyectos con Fechas, Presupuestos, Requisitos y Alcance fijo… “NO EXISTEN”



# Etapa 1: Análisis de Requisitos

**¿Qué es y en qué consiste?**

- Se refiere a obtener los requisitos de un producto de software a través de indagación de necesidades del cliente.

**¿Qué actividades comprende?**

- Entrevistas o comunicación con clientes o futuros usuarios para saber cuáles son sus expectativas.
- Detectar y corregir las carencias o falencias comunicativas. Retroalimentación.
- Documentar.
- Validar los requisitos obtenidos con el usuario.

**Problemas del Análisis de Requisitos**

- Los clientes piensan o creen que saben lo que el software debe hacer.
- Muchos requerimientos son dados de modo incompleto.
- Algunos requerimientos pueden ser contradictorios.

# Etapa 2: Especificación

**¿Qué es y en qué consiste?**

- Es la transformación del análisis en documentos que determinan el comportamiento esperado de las distintas funcionalidades y características inherentes del software.

**A considerar (respecto a funcionamiento)**

- Debe estar alineado 100% con las necesidades del negocio.
- Se debe determinar cómo será la interacción con los usuarios.

**Técnicas más usuales para plasmar especificaciones**

- Caso de uso ⇒ Métodos clásicos y formales.
- Historias de usuario ⇒ Métodos ágiles e informales (no tan informales).

**Lo que buscamos:**

- Definir el qué y el cómo del Software.

# Etapa 2: Especificación

**¿Cómo lo formalizamos y cómo sabemos que está bien?**

- Se formaliza a través del documento de SRS (Software Requirements Specification).

**Está bien realizado si el documento es:**

- Correcto ⇒ Los requerimientos deben estar en el SW.
- Inequívoco ⇒ Los requerimientos tienen solo una interpretación.
- Completo ⇒ Debe incluir:
  - *Los requisitos están relacionados a la funcionalidad, el desarrollo, las restricciones del diseño, los atributos y las interfaces externas.*
  - *La definición de las respuestas del software a todos los posibles datos de la entrada del sistema y a toda clase de situaciones.*
- Consistente ⇒ Ningún requerimiento se contradice.

# Etapa 2: Especificación

- Justificable ⇒ Logra delinear que tiene importancia y/o estabilidad. Esto implica:
  - *Cada requisito en él tiene un identificador para indicar su importancia o estabilidad en particular.*
  - *Saber que  los requisitos que relacionan a un producto del software no son igualmente importantes.*

- Comprobable ⇒ Debe tener condiciones concretas y medibles para saber si su implementación fue correcta.
- Modificable ⇒ Permite flexibilidad para cambios.
- Identificable ⇒ Su origen está claro y permite identificarlo antes, durante y después del desarrollo.

# Etapa 3: Diseño y Arquitectura

**¿Qué es y en qué consiste?**

- Es la definición de las distintas interacciones, capacidades y trazado de solución. Se refiere a:
  - *Como se llevará a cabo la integración de infraestructura.*
  - *Como se realizará el desarrollo de aplicaciones.*
  - *Como será el modelo y el uso bases de datos.*

**Habilidades requeridas de un arquitecto**

- Gran conocimiento de tecnología y el entorno.
- Capacidad de entendimiento y liderazgo.
- Visión de presente y futuro.

# Etapa 4: Programación

**¿Qué es y en qué consiste?**

- En palabras sencillas, es la codificación del Software en los distintos ambientes de un desarrollo.

**Objetivos explícitos**

- Realizar la programación del sistema.
- Lograr la interacción con los sistemas legados o interconectados.

**Objetivos implícitos**

- Optimizar las líneas y estructura del código.
- Lograr que se respeten los plazos de programación.
- Obtener la menor cantidad de “bugs” posibles.
- Realizar pruebas unitarias.

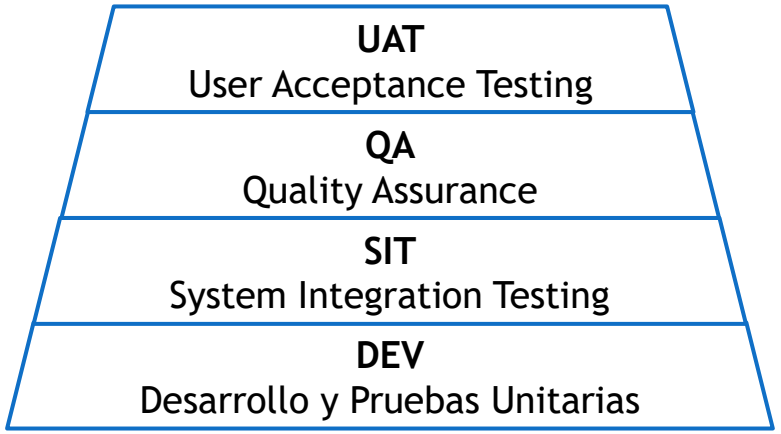
# Etapa 5: Pruebas

**¿Qué es y en qué consiste?**

- En esta etapa, interconectada a distintas fases, comprobamos que el software hace las tareas que se definieron en su especificación y cumple con sus parámetros no funcionales.

**Tipos de pruebas**

- Pruebas Unitarias
- Pruebas de Modulares o de Integración
- Pruebas de Sistema
- Pruebas Interconectadas




**Recomendaciones**

- Las pruebas debe realizarlas, por lo general, un equipo destinado a ello: QA.
- El equipo de QA debe ser formado por personas no relacionadas a la etapa de desarrollo.

# Etapa 6: Documentación

**¿Qué es y en qué consiste?**

- Se refiere a la recopilación de TODO lo realizado como parte del proceso de desarrollo de SW.

**Tipos de documentos**

- Documento de Requerimientos.
- Documento de Gestión del Proyecto.
- Documento de Especificación (con diagramas UML).
- Documento de Diseño y Arquitectura (con diagramas UML).
- Documento de Pruebas y Resultados.
- Documento de paso a Producción (Runbook).
- Manuales Técnicos y de Usuario.

# Etapa 7: Mantenimiento

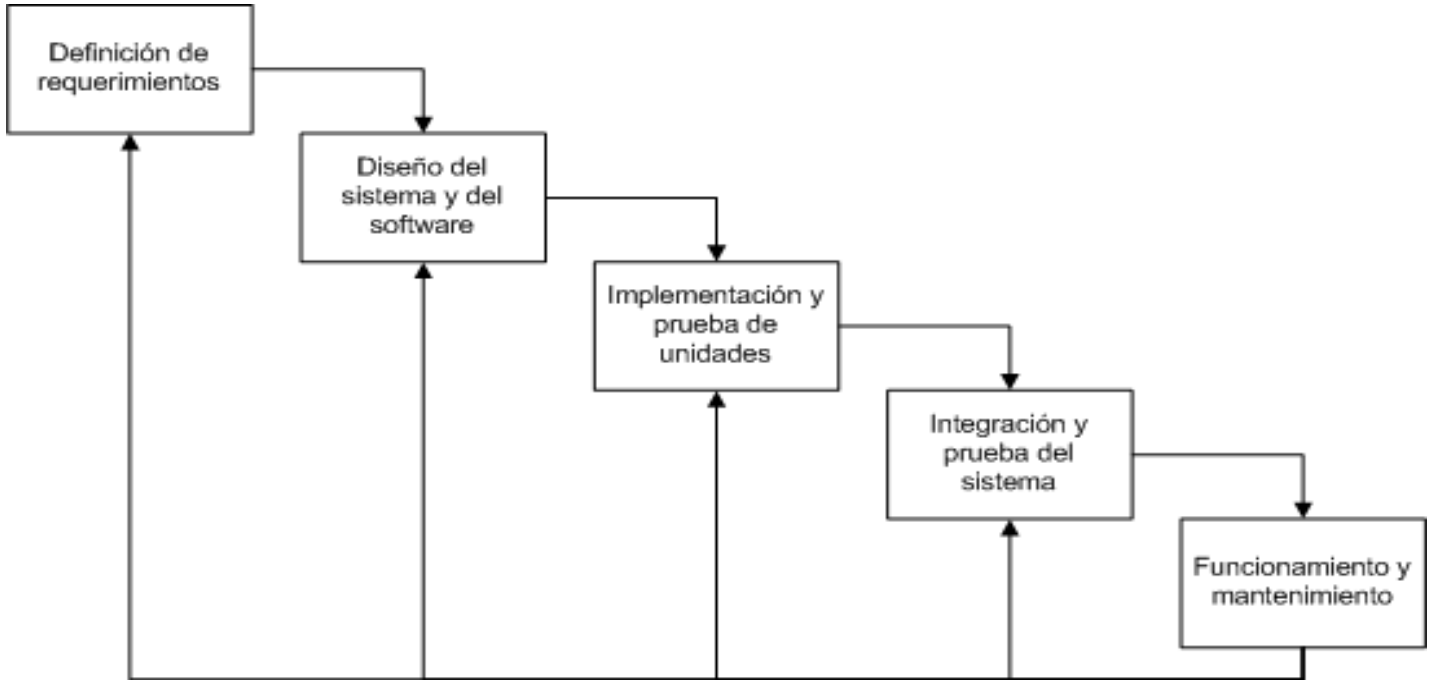
**¿Qué es y en qué consiste?**

- Es la fase que, cuando el SW está en producción, se ocupa de:
  - *Mantenimiento del Sistema o Preventivo ⇒ Operatividad.*
  - *Mantenimiento adaptativo o Mejoras al sistema ⇒ Optimización.*
  - *Mantenimiento evolutivo o Adiciones al sistema ⇒ Cambios obligatorios.*
  - *Mantenimiento Correctivo o eliminación de errores ⇒ Eliminación de bugs.*

# MODELOS DE DESARROLLO DE SOFTWARE

# Modelo de Desarrollo en Cascada

- Es un modelo de desarrollo muy riguroso que ordena las etapas de modo tal que una etapa no comienza sino hasta que termina la anterior.
- Considera las actividades fundamentales del proceso de especificación, desarrollo, validación y evolución, y los representa como fases separadas del proceso.



# Modelo de Desarrollo en Cascada

**Modelo en Cascada - Etapa 1: Definición de Requerimientos**

Los requerimientos son los servicios, restricciones y metas del sistema que se definen a partir de las consultas con los usuarios. Entonces, se detallan y sirven como una especificación del sistema.

Respondamos las siguientes preguntas:

- *¿Qué son los Servicios del Sistema?*
- *¿Qué son las restricciones del Sistema?*
- *¿Qué son las Metas del sistema?*

# Modelo de Desarrollo en Cascada

**Modelo en Cascada - Etapa 2: Diseño del Sistema y del Software**

Es el proceso que:

- Divide los requerimientos en sistemas de hardware o software.
- Establece una arquitectura completa del sistema.
- Identifica y describe las abstracciones fundamentales del sistema software y sus relaciones.

**Modelo en Cascada - Etapa 3: Implementación y Prueba de unidades**

Es llevar el diseño del software, a través de programación, a un conjunto o unidades de programas. La prueba de unidades implica verificar que cada una cumpla su especificación.

Respondamos las siguientes preguntas:

- *Si las pruebas unitarias cumplen su especificación ¿El SW funcionará adecuadamente? ¿Por qué?*
- *Verificar y Validar ¿Son lo mismo?*

# Modelo de Desarrollo en Cascada

**Modelo en Cascada - Etapa 4: Integración y Pruebas de Sistema**

- Los programas o las unidades individuales de programas se integran y
- se prueban como un sistema completo para asegurar que se cumplan los requerimientos del software.

Después de las pruebas, el sistema software se entrega al cliente.

Respondamos las siguientes preguntas:

- *¿Qué significa Integración?*
- *¿Qué es la prueba de Integración?*
- *¿Qué es la prueba de Sistema?*
- *¿Qué unidad de un equipo de desarrollo realiza estas pruebas?*

# Modelo de Desarrollo en Cascada

**Modelo en Cascada - Etapa 5: Funcionamiento y Mantenimiento**

- Por lo general (aunque no necesariamente), ésta es la fase más larga del ciclo de vida. El sistema se instala y se pone en funcionamiento práctico. El mantenimiento implica:
- Corregir errores no descubiertos en las etapas anteriores del ciclo de vida.
- Mejorar la implementación de las unidades del sistema y resaltar los servicios del sistema una vez que se descubren nuevos requerimientos.

Respondamos las siguientes preguntas:

- *¿Por qué se le conoce como la etapa más larga del Ciclo de Vida del Software?*
- *¿Es un error que se descubran nuevos requerimientos?*
- *¿Qué implica corregir errores?*

# Modelo de Desarrollo en Cascada

**Modelo en Cascada - Ventajas**

- Se tiene todo bien organizado y no se mezclan las fases.
- Es perfecto para proyectos que son rígidos, y además donde se especifiquen muy bien los requerimientos y se conozca muy bien la herramienta a utilizar

**Modelo en Cascada - Desventajas**

- En la vida real, un proyecto rara vez sigue una secuencia lineal.
- Su mala implementación lleva al fracaso.

**Modelo en Cascada - Consideraciones**

- En teoría, una fase no debe empezar hasta finalizar la fase previa, pero en la práctica, las fases se superponen y entregan información a las otras.
- Al evolucionar el sistema, pueden aparecer nuevos requerimientos, por lo que se pueden repetir partes del proceso.
- Este modelo solo se debe utilizar cuando los requerimientos se comprendan bien y sea improbable que cambien durante el desarrollo.

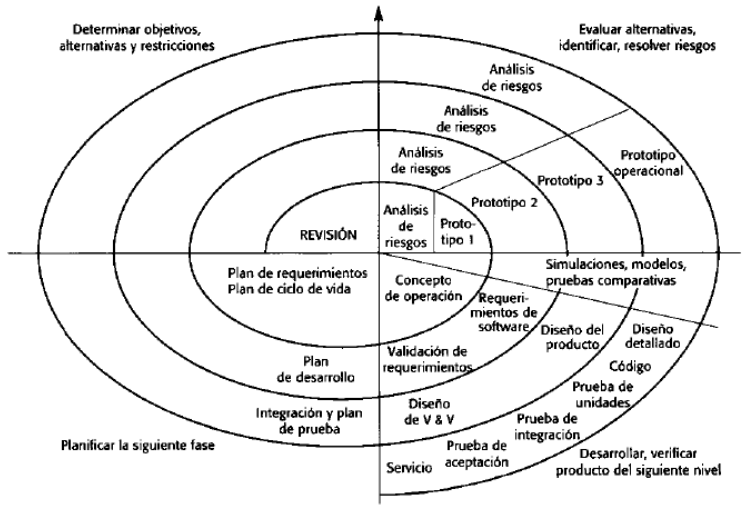
¿Para qué tipo de Software podría utilizar este Modelo de Desarrollo?

# Modelo de Desarrollo en Espiral

- Es un modelo donde más que representar el proceso del software como una secuencia de actividades con retrospectiva de una actividad a otra, se representa como una espiral.
- Cada ciclo en la espiral representa una fase del proceso del software. Así, el ciclo más interno se podría referir a la viabilidad del sistema, el siguiente ciclo a la definición de requerimientos, el siguiente ciclo al diseño del sistema, y así sucesivamente.

Además, cada ciclo de la espiral se divide en cuatro sectores:

- Definición de Objetivos
- Evaluación y Reducción de Riesgos
- Desarrollo y Validación
- Planificación



# Modelo de Desarrollo en Espiral

**Fase 1: Definición de Objetivos**

¿Qué hacemos en esta fase?

- Se definen los objetivos específicos.
- Se identifican las restricciones del proceso y el producto
- Se traza un plan detallado de gestión.
- Se identifican los riesgos del proyecto.
- Entendiendo estos riesgos se planean estrategias alternativas.

***Respondamos:***

- ¿Cada nueva espiral tendrá nuevos objetivos?
- ¿Es relevante el riesgo para determinar el orden de resolución de objetivos?

# Modelo de Desarrollo en Espiral

**Fase 2: Evaluación y Reducción de Riesgos**

Aquí se lleva a cabo un análisis detallado para cada riesgo identificado y se definen los pasos para reducir dichos riesgos.

***Ejemplo:***

Si existe el riesgo de tener requerimientos inapropiados, se puede desarrollar un prototipo del sistema.

***Respondamos:***

¿Cada nueva espiral tendrá nuevos análisis de riesgo?

¿Cómo podemos determinar la mejor manera de reducir cada riesgo?

# Modelo de Desarrollo en Espiral

**Fase 3: Desarrollo y Validación**

Después de la evaluación de riesgos, se elige un modelo para el desarrollo del sistema.

***Ejemplo:***

- Si los riesgos en la interfaz de usuario son dominantes, un modelo de desarrollo apropiado podría ser la construcción de prototipos evolutivos.
- Si los riesgos de seguridad son la principal consideración, un desarrollo basado en transformaciones formales podría ser el más apropiado, y así sucesivamente.

***Respondamos:***

¿Por qué el modelo en cascada podría ser más apropiado para el desarrollo si el mayor riesgo identificado es la integración de los subsistemas?

# Modelo de Desarrollo en Espiral

**Fase 4: Planificación**

En la planificación se revisa el proyecto y se toma la decisión de si se debe continuar con un ciclo posterior de la espiral. Si se decide continuar, se desarrollan los planes para la siguiente fase del proyecto.

***Respondamos:***

- ¿Cómo sabemos que debemos realizar un nuevo ciclo?
- El modelo en cascada ¿Consideraba el riesgo de manera explícita?

# Modelo de Desarrollo en Espiral

**Ventajas del Modelo**

- El análisis del riesgo se hace de forma explícita y clara.
- Incorpora objetivos de calidad.
- Integra el desarrollo con el mantenimiento.
- Además, es posible tener en cuenta mejoras y nuevos requerimientos sin romper con la metodología.

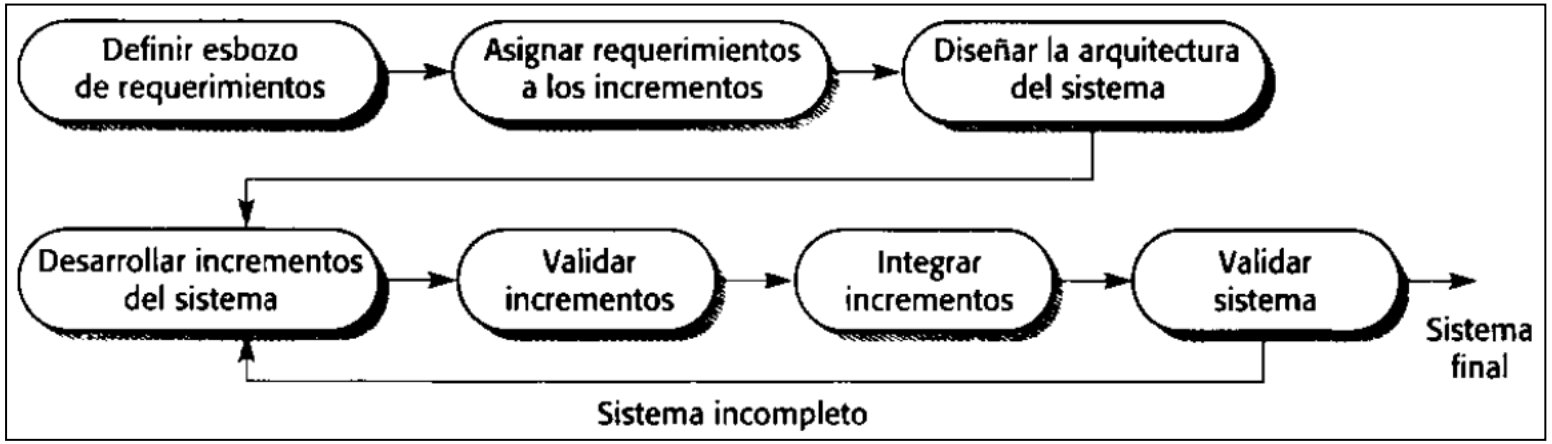
**Desventajas del Modelo**

- Genera mucho tiempo en el desarrollo del sistema.
- El desarrollo puede tornarse muy costoso.
- Requiere experiencia en la identificación de riesgos.

# Modelo de Desarrollo Iterativo-Incremental

Es un modelo de desarrollo donde la especificación, el diseño y la implementación del software se dividen en una serie de incrementos, los cuales se desarrollan por turnos.

- En el enfoque incremental, no existe una especificación completa del sistema hasta que el incremento final se especifica.



# Modelo de Desarrollo Iterativo-Incremental

**Modelo Iterativo/Incremental – Etapa 1: Definir el esbozo de Requerimientos**

- Son los servicios, restricciones y metas del sistema que se definen a partir de las consultas con los usuarios. Entonces, se detallan y sirven como una especificación del sistema.

**Modelo Iterativo/Incremental – Etapa 2: Asignar requerimientos a los incrementos**

Es el proceso que:

- Se separan los requerimientos y se forman conjuntos interrelacionados.
- Se priorizan de acuerdo a algún parámetro.
- Se agrupan y asocian a un “número” de incremento particular

**Modelo Iterativo/Incremental – Etapa 3: Diseñar la Arquitectura del Sistema**

- Es llevar el cúmulo de incrementos del software a un estudio de necesidades y, finalmente, definir sus interacciones, tecnología para soportarlos y estándares de desarrollo.

# Modelo de Desarrollo Iterativo-Incremental

**Modelo Iterativo/Incremental – Etapa 4, Cíclica: Desarrollar los incrementos**

- Es llevar el diseño del software, a través de programación, a un conjunto o unidades de programas para cumplir con lo que se debe construir en cada iteración.

**Modelo Iterativo/Incremental – Etapa 5, Cíclica: Validar el incremento**

- Es revisar, idealmente junto al usuario y/o cliente, que el incremento cumple sus expectativas.

**Modelo Iterativo/Incremental – Etapa 6, Cíclica: Integrar los incrementos**

- Es reunir el incremento a los otros incrementos.

**Modelo Iterativo/Incremental – Etapa 7, Cíclica/Final: Validar el Sistema**

- Es reunir el conjunto de incrementos al sistema completo que “interactúa” con él.

# Modelo de Desarrollo Iterativo-Incremental

**Modelo Iterativo/Incremental - Ventajas**

- Se reduce el tiempo de desarrollo inicial, ya que se implementa la funcionalidad parcial.
- Tiene un impacto frente al cliente por la entrega temprana de partes operativas del software.
- Proporciona todas las ventajas del modelo en Cascada realimentado, reduciendo sus desventajas sólo al ámbito de cada incremento.
- Resulta más fácil acomodar cambios al acotar el tamaño de los incrementos.

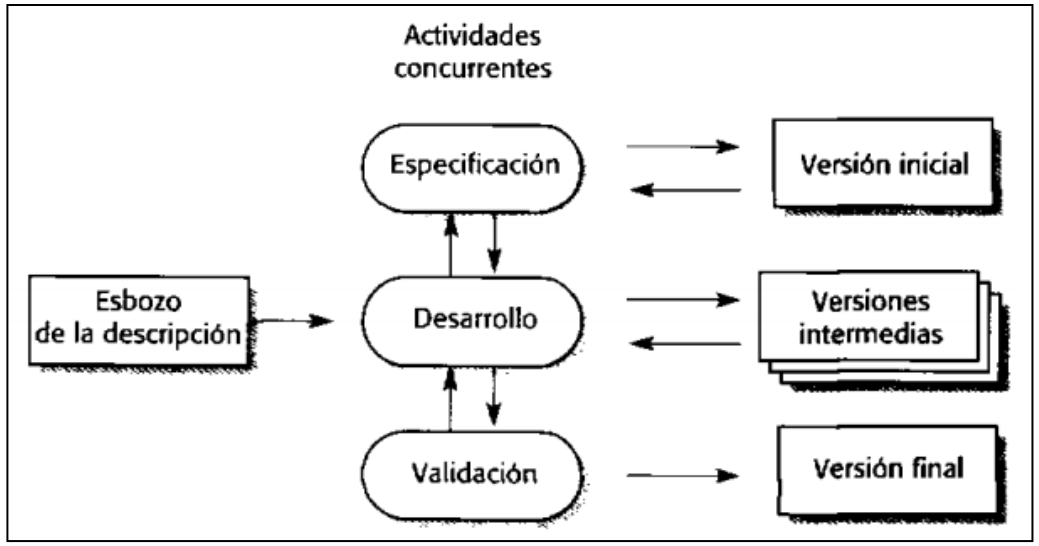
**Modelo Iterativo/Incremental - Desventajas**

- El modelo incremental no es recomendable para casos de sistemas de tiempo real, de alto nivel de seguridad, de procesamiento distribuido y/o de alto índice de riesgos.
- Requiere de mucha planeación y necesita metas claras para conocer el estado del proyecto.

# Modelo de Desarrollo por Etapas

Es un modelo del tipo “evolutivo” que considera que los requerimientos no son del todo conocidos al comenzar el proyecto.  Esto implica:

- Se van desarrollando las especificaciones a medida que se está trabajando en una versión.
- Existen actividades recurrentes requieren interacción directa con el cliente para el feedback.



# Modelo de Desarrollo por Etapas

**¿Qué ocurre en la práctica?**

La especificación, desarrollo y validación se entrelazan en vez de separarse. Así se logra una rápida retroalimentación entre estas actividades. Este modelo (al igual que otros modelos evolutivos) comprende dos tipos de desarrollo

**Desarrollo exploratorio**

- Es donde se trabaja con el cliente para explorar sus requerimientos y entregar un sistema final.
- El desarrollo empieza con las partes del sistema que se comprenden mejor.
- El sistema evoluciona agregando nuevos atributos propuestos por el cliente.

**Prototipos desechables**

- Aquí se busca comprender los requerimientos del cliente y entonces desarrollar una definición mejorada de los requerimientos para el sistema.
- El prototipo se centra en experimentar con los requerimientos del cliente que no se comprenden del todo.

# Modelo de Desarrollo por Etapas

**Ventajas del Modelo**

- Se realiza un constante intercambio de información con el cliente.
- Se realizan entregas periódicas del Sistema.
- Se priorizan las necesidades del negocio de acuerdo a cómo las enfrenta el cliente.

**Desventajas del Modelo**

- No facilita la integración de aplicaciones que han sido desarrolladas como sistemas independientes.
- Puede generar "esclerosis de información“. Es decir, algunas deficiencias del software se hacen inmodificables junto a la evolución. Luego, no será posible seguir evolucionando.

# METODOLOGÍAS DE DESARROLLO DE SOFTWARE

# Metodologías de Desarrollo ágil de SW

Las metodologías ágiles son aquellas que permiten adaptar la forma de trabajo a las condiciones del proyecto, consiguiendo flexibilidad e inmediatez en la respuesta para amoldar el proyecto y su desarrollo a las circunstancias específicas del entorno.

**Su apuesta**

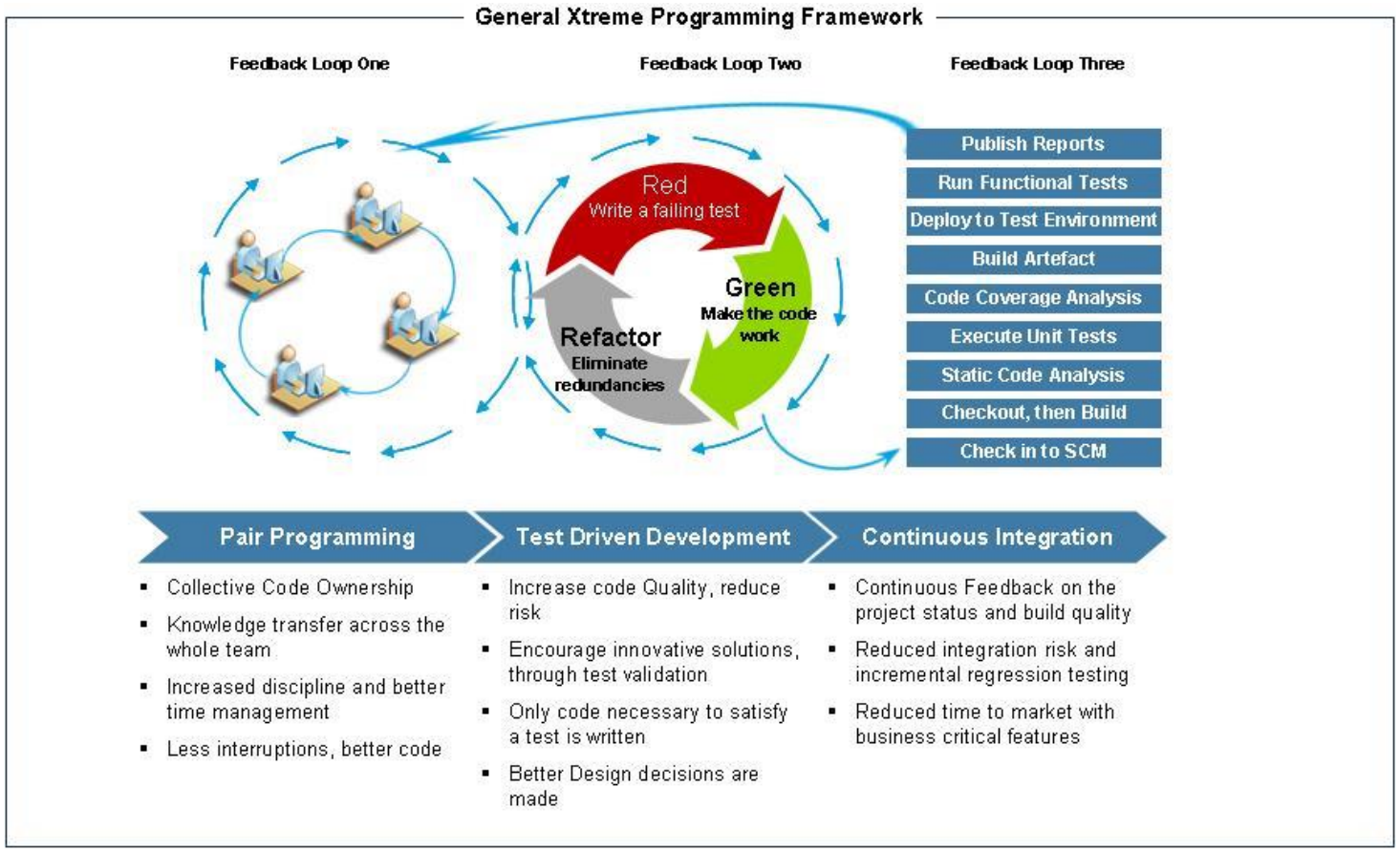
Entregar valor de manera constantemente

| SPRINT 1 | SPRINT 2 | SPRINT 3 | SPRINT 4 | SPRINT 5 | SPRINT 6 |
|---|---|---|---|---|---|
|  |  |  |  |  |  |
| Entretención Feedback | Nuevas funcionalidades | Mejora de funcionalidad | Innovación rechazada | Innovación aceptada | Escalabilidad |

***Ejemplo adaptado de:*** *https://mamaqueesscrum.com/2019/04/15/construyendo-un-columpio-con-scrum-dividiendo-el-alcance-en-trozos-con-valor/*

# Programación Extrema (XP)

XP es una metodología de desarrollo de software que está destinada a mejorar la calidad del software y capacidad de respuesta a los cambios en los requisitos del cliente.



# Programación Extrema (XP)

**Requisitos e Historias de usuario**

- Una historia de usuario es una representación de un requerimiento de software escrito en frases simples y utilizando el lenguaje usual del usuario (lenguaje natural).
- Una historia de usuario puede tener una o más tareas asociadas que, al ser desarrolladas, permiten cumplir con el conjunto de requisitos en la historia.

**Descarga e impresión de un artículo**

En primer lugar, seleccione el artículo que desea de una lista visualizada. Tiene entonces que decirle al sistema cómo lo pagará —se puede hacer a través de una suscripción, una cuenta de empresa o mediante una tarjeta de crédito.

Después de esto, obtiene un formulario de derechos de autor del sistema para que lo rellene. Cuando lo haya enviado, se descarga el artículo en su computadora.

Elija una impresora y se imprimirá una copia del artículo. Le dice al sistema que la impresión se ha realizado correctamente.

Si es un artículo de sólo impresión, no puede guardar la versión en PDF, por lo que automáticamente se elimina de su computadora.

**Tarea 1: Implementar el flujo de trabajo principal**

**Tarea 2: Implementar catálogo y selección de artículos**

**Tarea 3: Implementar formas de pago**

El pago se puede efectuar de 3 formas diferentes. El usuario selecciona de qué forma desea pagar. Si el usuario tiene una suscripción a la biblioteca, puede introducir la clave de suscriptor, la cual debe ser verificada por el sistema. De forma alternativa, puede introducir un número de cuenta organizacional. Si es válido, se anota un cargo en la cuenta por el importe del artículo. Finalmente, puede introducir un número de tarjeta de crédito de 16 dígitos y la fecha en la que caduca. Se debe comprobar la validez de estos datos y, si son válidos, se anota un cargo en la cuenta de la tarjeta de crédito.

# Scrum

Método ágil basado en el Modelo Iterativo/Incremental que se enfoca en la gestión de procesos de desarrollo de software y que propone un conjunto de prácticas y roles para definir el proceso de desarrollo que se ejecutará durante un proyecto.

# Scrum: Ejemplo práctico

Paso 1: Un cliente se pone en contacto con una empresa que fabrica Robots y realiza el pedido.

Paso 2: El cliente se reúne con el Product Owner quien toma nota de lo que desea (tiene pensado) el cliente


*Ejemplo creado por Daniel Escribano Alés, CEO de Crowdslides*

# Scrum: Ejemplo práctico

Paso 3: El producto se divide en historias (partes o entregables) que definen la Pila del Producto

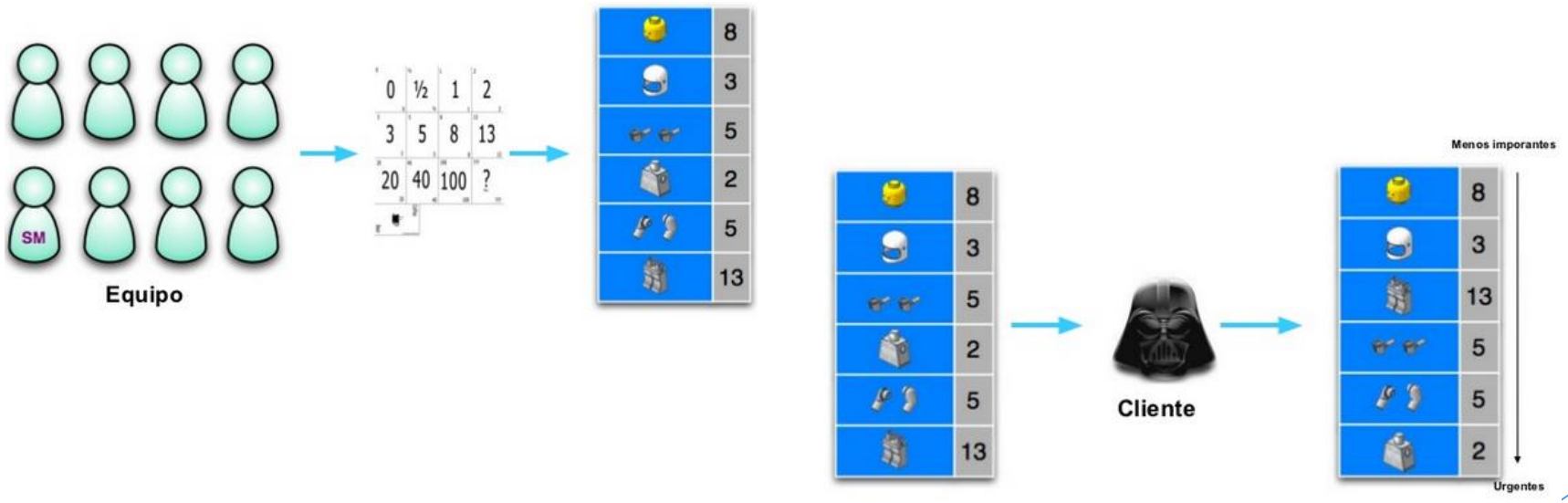
Paso 4: El Product Owner se reúne con el ScrumMaster para entregar la Pila de Producto


*Ejemplo creado por Daniel Escribano Alés, CEO de Crowdslides*

# Scrum: Ejemplo práctico

Paso 5: El equipo se reúne para estimar el costo (esfuerzo) de capa historia de la pila.

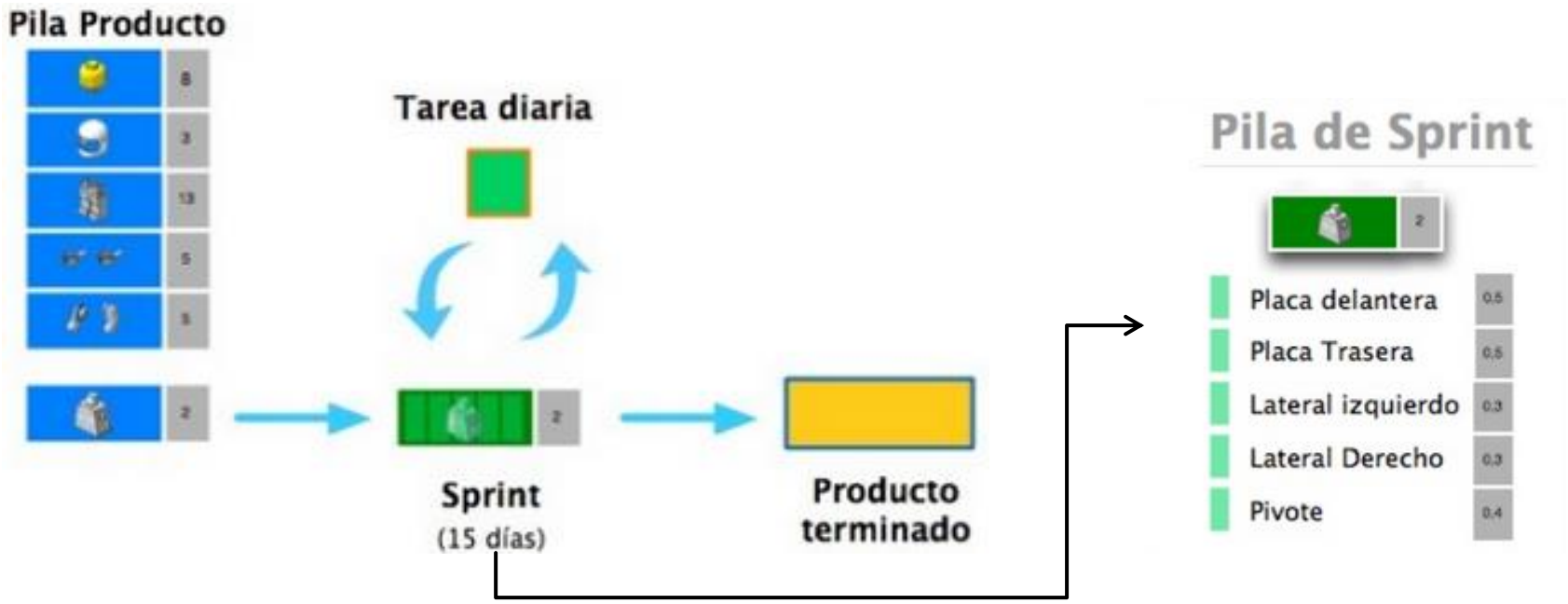
Paso 6: El cliente, una vez aprobado el presupuesto, reordena la pila de acuerdo a sus prioridades para el desarrollo (de mayor a menor o menor a mayor urgencia, pero con claridad).




*Ejemplo creado por Daniel Escribano Alés, CEO de Crowdslides*

# Scrum: Ejemplo práctico

Paso 7: El equipo comienza su trabajo desglosando la primera historia en tareas menores para crear el Sprint y su pila respectiva.

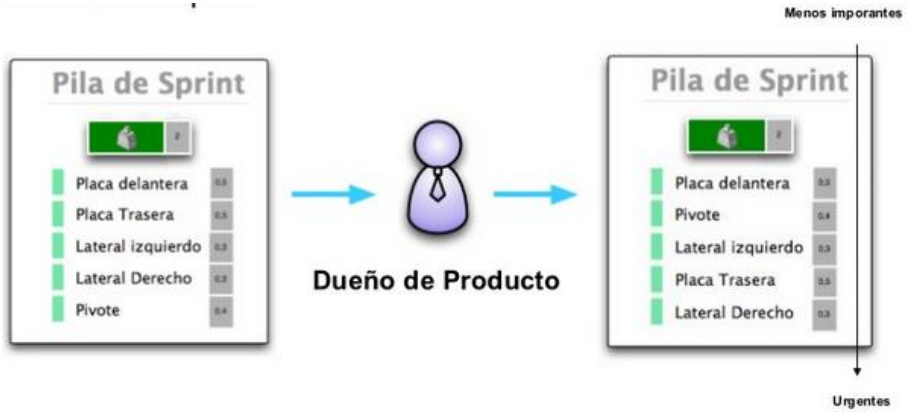



*Ejemplo creado por Daniel Escribano Alés, CEO de Crowdslides*

# Scrum: Ejemplo práctico

Paso 8: El Product Owner, después de consultar al cliente, prioriza las tareas antes de comenzar el Sprint

Paso 9: El equipo comienza el Sprint tomando las tareas priorizadas. Una vez terminada una tarea, se toma la siguiente. Así mismo, a diario se generan reuniones para ver el avance del equipo.



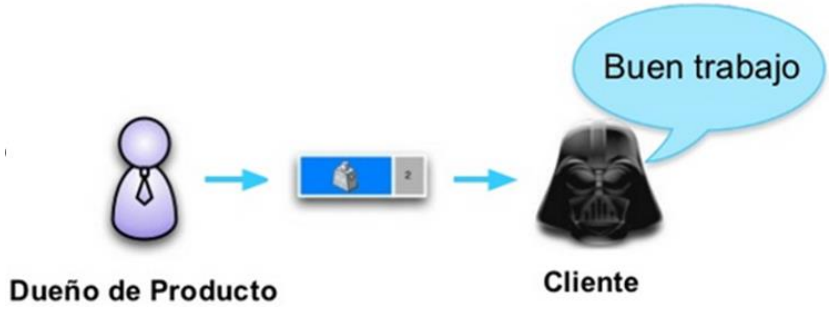


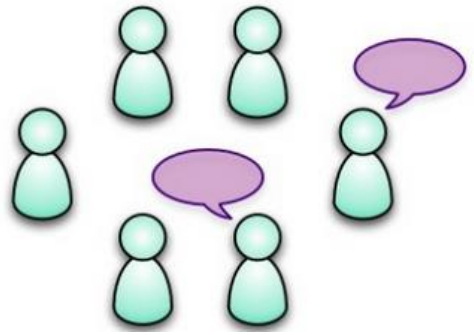
*Ejemplo creado por Daniel Escribano Alés, CEO de Crowdslides*

# Scrum: Ejemplo práctico

Paso 10: Al finalizar el sprint, el Product Owner se reúne con el cliente para mostrar el avance. Con ello, el cliente tiene la primera entrega y puede, si lo desea, reordenar la Pila para que comience otro Sprint.

Paso 11: El equipo se reúne para hacer la revisión del Sprint y su retrospectiva. Se recomienda un ambiente distendido (fuera de la oficina) para esta reunión.



*Ejemplo creado por Daniel Escribano Alés, CEO de Crowdslides*

# Cierre de la sesión