Programación 2

Programas, software,
ciclo de vida

Programas

En programación 1 vimos que podemos resolver problemas bajo el paradigma imperativo, donde el foco está en las instrucciones:

> *"un **programa** es una **secuencia de instrucciones** escritas en un **lenguaje** de programación, que permiten **resolver un problema**"*

Sistemas

Los programas son componentes de software que se desarrollan como parte de un sistema mayor que resuelve un problema.

Los problemas complejos requieren la construcción de **sistemas** que incluyen componentes de **software**.

Software

El **software** está formado por un conjunto de **programas** y los **documentos** que modelan el problema y su solución.

Siempre se espera que los sistemas sean **confiables**, **eficientes** y **flexibles** independientemente de su escala, y funcionando en **ambientes dinámicos**.

La escala de un problema está relacionada con el costo y el tiempo que demanda resolverlo.

Un ambiente dinámico es aquel en el que se producen cambios frecuentes.

Desarrollo de software

El desarrollo de software es un **proceso** que abarca distintas etapas y requiere de la aplicación de una metodología.

Es un proceso **colaborativo** en el que interactúan los miembros del equipo de desarrollo con clientes y usuarios.

El **ciclo de vida** de un sistema de software comienza cuando se formula la necesidad, oportunidad o idea que le da origen y termina cuando deja de utilizarse.

Las **etapas del desarrollo** pueden organizarse de diferentes maneras, una alternativa es el modelo en cascada.

Ciclo de vida en cascada

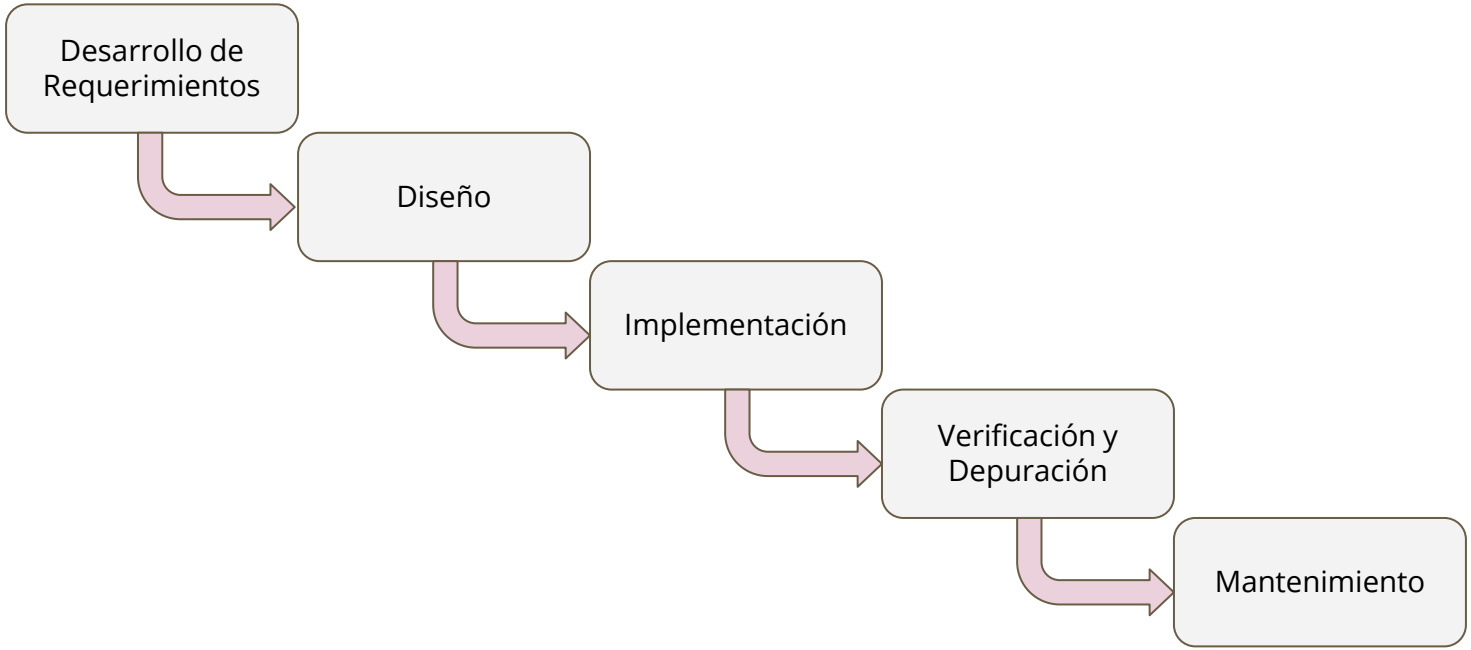
El ciclo de vida en cascada propone un **orden** específico para establecer y ordenar las **etapas** del proceso de desarrollo de software.

Las etapas conforman una **secuencia**, en la cual el resultado de una etapa es la entrada de la etapa siguiente.

En el proceso participan distintos participantes, entre ellos:

- El cliente (que demanda el producto)
- El equipo de profesionales que lo desarrolla
- Los usuarios (que finalmente lo van a utilizar, son quienes van a interactuar con el sistema)

Ciclo de vida en cascada



Desarrollo de los requerimientos

El desarrollo del sistema surge para satisfacer una demanda existente, que puede surgir de una necesidad, una oportunidad o una idea.

Si el sistema no satisface la demanda, entonces no tendrá éxito.

En esta etapa sólo se genera **documentación** (especificación de los requerimientos) donde se establece:

- **Qué** problema hay que resolver
- **Qué** características debe tener la solución y qué restricciones debe cumplir
- **Por qué** es un problema y por lo tanto requiere solución
- **Quienes** serán responsables de participar en el desarrollo de la solución

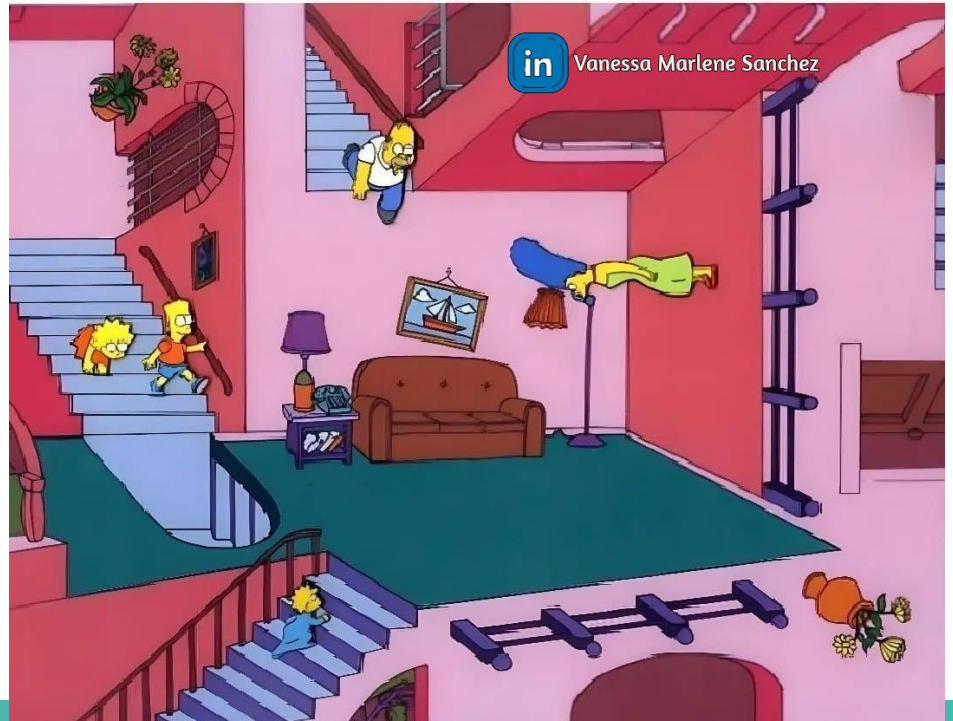
Diseño

A partir de la documentación de los requerimientos se **diseña una solución** para el problema especificado.

El resultado de esta etapa es un **documento** elaborado por uno o más diseñadores, que describen los módulos que integrarán el sistema y la forma en que se relacionan entre sí. El documento también puede establecer casos de prueba o tests que se aplicarán en la etapa de verificación.

Importancia de la documentación

El programador/a intentando entender el código de un proyecto mal documentado



Implementación

A partir del documento generado en la etapa de diseño, los desarrolladores o programadores generan el **programa** escrito en un lenguaje de programación y toda la documentación referida al código.

Es importante que el programa implementado mantenga la estructura especificada en la etapa de diseño.

Cada **módulo** del diseño se debe corresponder con una **unidad de código implementada**, con cierta independencia del sistema completo.

Verificación y depuración

Verificación: evaluar si el sistema satisface la especificación de requerimientos.

Depuración: corregir los errores que se detectan.

Cuando el sistema está dividido en módulos, cada unidad de código se verifica y depura por separado, y luego se verifica y depura la integración de los módulos.

Los casos de prueba se pueden haber establecido en la etapa de diseño, o se pueden definir en esta etapa.

Es importante que al menos una parte de la verificación la realicen personas que no participaron en la implementación.

Observación: el término validación (que pueden encontrar en la bibliografía) se refiere a la evaluación del sistema en función de las necesidades reales de los usuarios, que en ocasiones no se corresponde con los requerimientos especificados. En este curso asumimos que los requerimientos de los problemas propuestos corresponden a las necesidades de los usuarios.

Mantenimiento

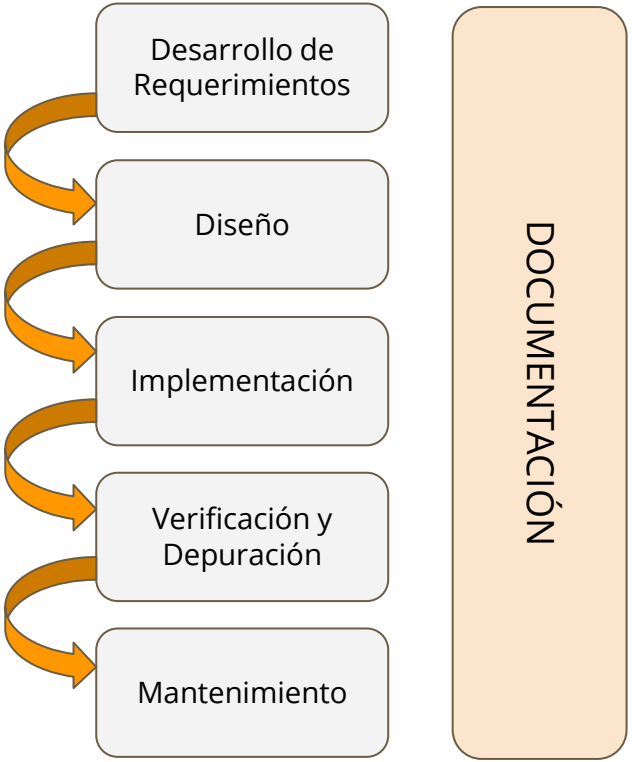
Es normal que cambien las necesidades del usuario durante el transcurso del ciclo de vida de un sistema de software. (en general, crecen las necesidades).

El mantenimiento involucra todos los cambios en el software que resultan de cambios en las especificaciones de requerimientos.

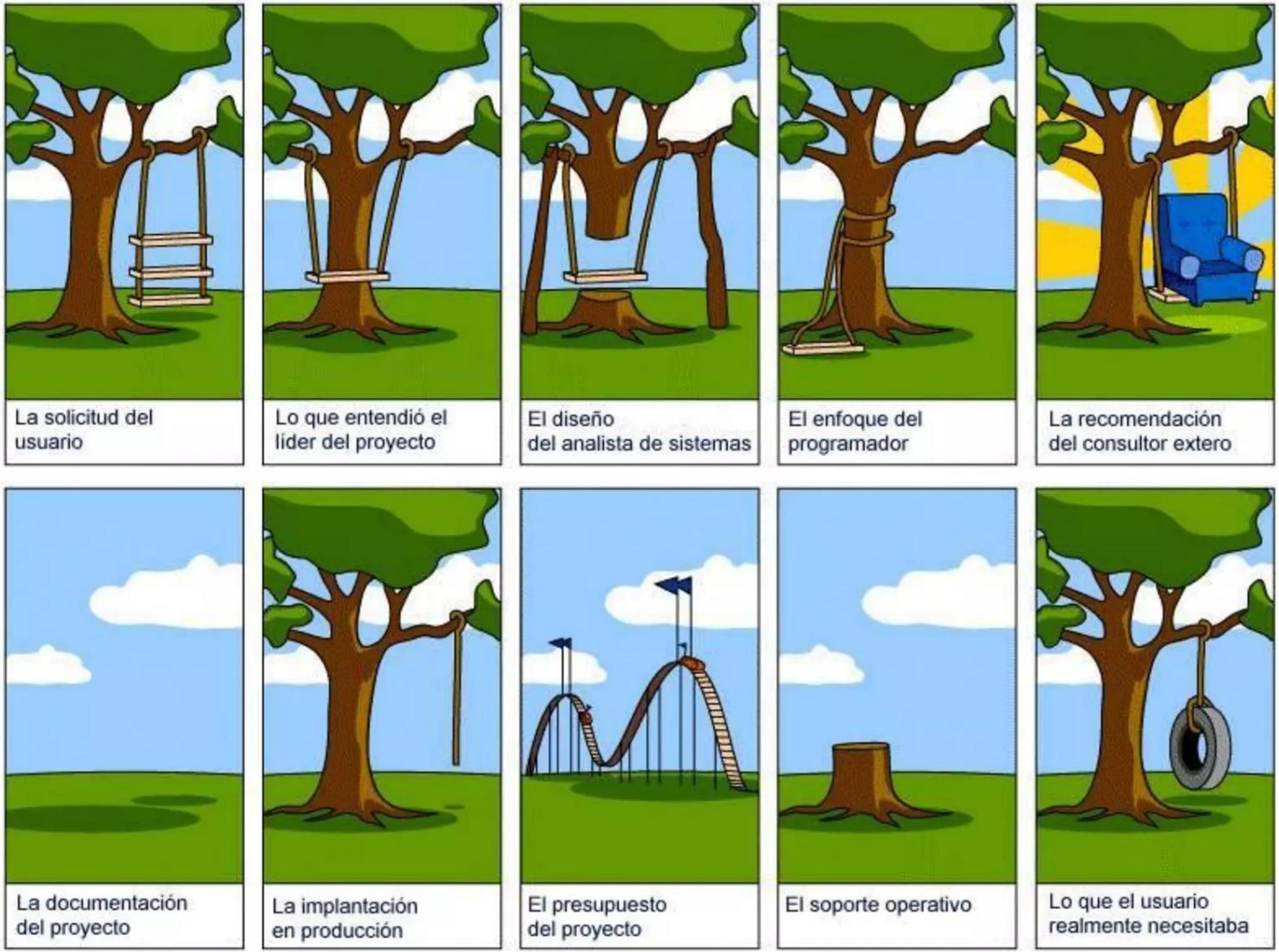
El diseño modular y la documentación resultan fundamental para el control del impacto de los cambios.

En un sistema bien modulado los cambios menores impactan sobre un conjunto reducido de módulos, o incluso pueden provocar la necesidad de generar nuevos módulos sin afectar a los existentes.

Ciclo de vida en cascada



Importancia de la documentación rigurosa, y consecuencias de la falta de documentación.



Desarrollo ágil

El desarrollo ágil se refiere a una **estrategia** de desarrollo de software colaborativo y multi-funcional, en la que las soluciones se crean bajo un **método rápido y flexible** para lograr el resultado deseado

- **Iterativo**: El desarrollo ágil se divide en pequeñas iteraciones, llamadas sprints, que suelen durar entre una y cuatro semanas.
- **Flexibilidad**: El equipo de desarrollo ágil se adapta rápidamente a los cambios en los requisitos y prioriza las tareas según sea necesario.
- **Colaboración**: Los miembros del equipo trabajan juntos de manera multidisciplinaria, con responsabilidades compartidas y comunicación abierta.
- **Priorización**: El trabajo pendiente se prioriza según su importancia y se entrega en pequeñas porciones de funcionalidad.
- **Retroalimentación**: El equipo recibe retroalimentación continua del cliente y ajusta el curso del proyecto según sea necesario.

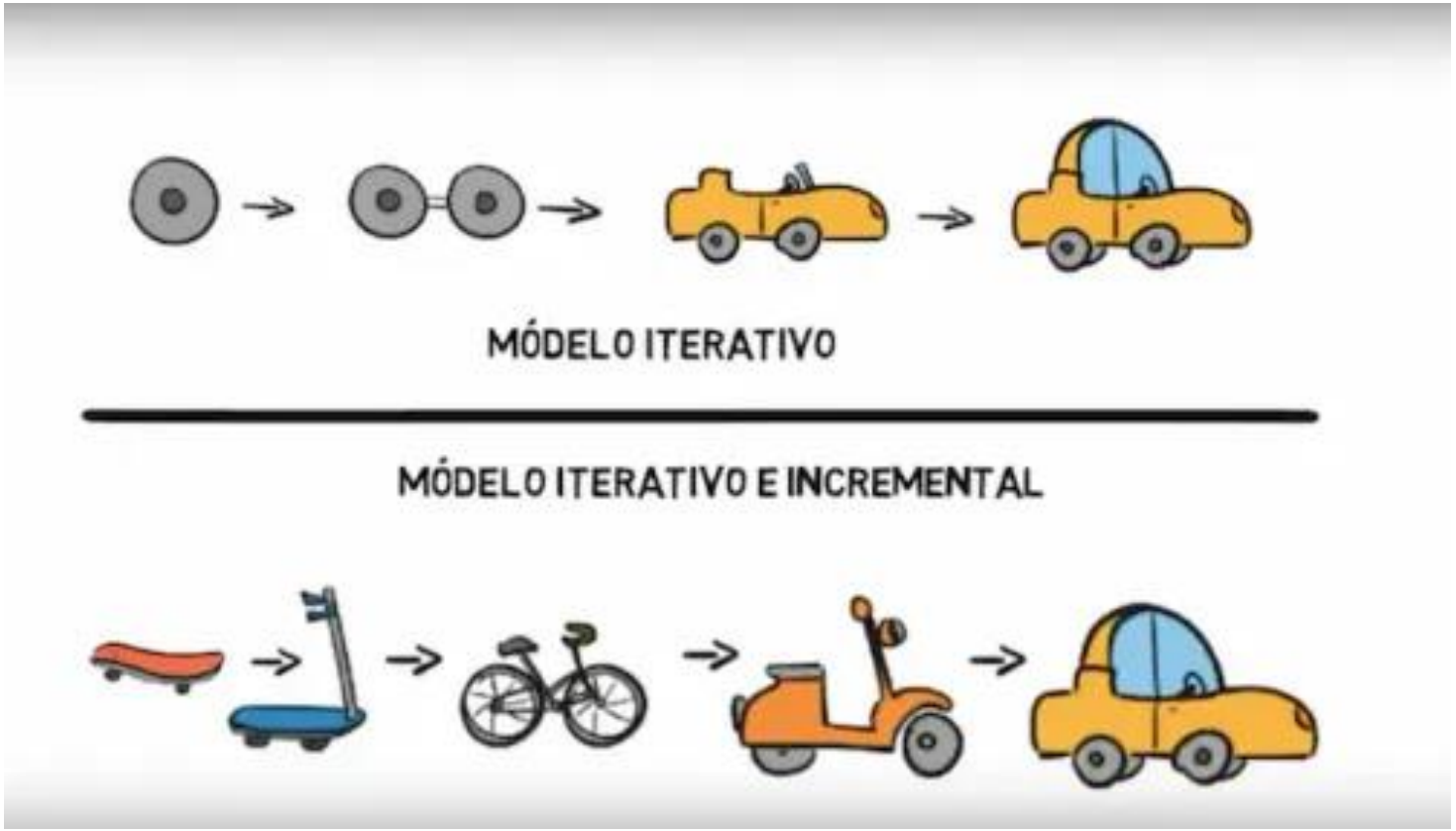
Desarrollo ágil - Beneficios

Entrega rápida: El desarrollo ágil permite entregar productos funcionales y valiosos en un plazo corto.

Flexibilidad: El equipo puede adaptarse rápidamente a los cambios en los requisitos y priorizar las tareas según sea necesario.

Mejora continua: El desarrollo ágil fomenta la mejora continua y la retroalimentación, lo que conduce a productos de alta calidad.

Colaboración y comunicación: El desarrollo ágil promueve la colaboración y comunicación entre los miembros del equipo y el cliente.


MÓDELO ITERATIVO
MÓDELO ITERATIVO E INCREMENTAL

Desarrollo ágil - Scrum

Scrum es una **metodología ágil** de gestión de proyectos que permite dividir a los equipos para estructurar y organizar el trabajo basándose en una serie de reglas, principios y prácticas. El término fue publicado por primera vez en la revista Harvard Business Review en 1986, en un artículo firmado por los japoneses Hirotaka Takeuchi e Ikujiro Nonaka.

Estos tomaron como referencia dos enfoques:

- La primera nos traslada a las empresas manufactureras de la industria automotriz, donde se empleaban metodologías para aumentar la velocidad y flexibilidad (como por ejemplo Lean).
- La otra la encontramos en el rugby, de donde los autores se inspiraron fuertemente: “Para recorrer una distancia determinada y marcar un tanto, un equipo multidisciplinar debe trabajar como una unidad, pasándose la pelota de un lado al otro lo más rápido posible y superando los obstáculos que aparecen en el camino”.

A diferencia de las metodologías tradicionales, con la que se trabaja de **forma secuencial**, Scrum apuesta por el desarrollo de **producto incremental y por la entrega continua de valor**. Esto permite adaptarse mejor a los cambios, resolver errores más rápidamente y aumentar la satisfacción del cliente.

Scrum

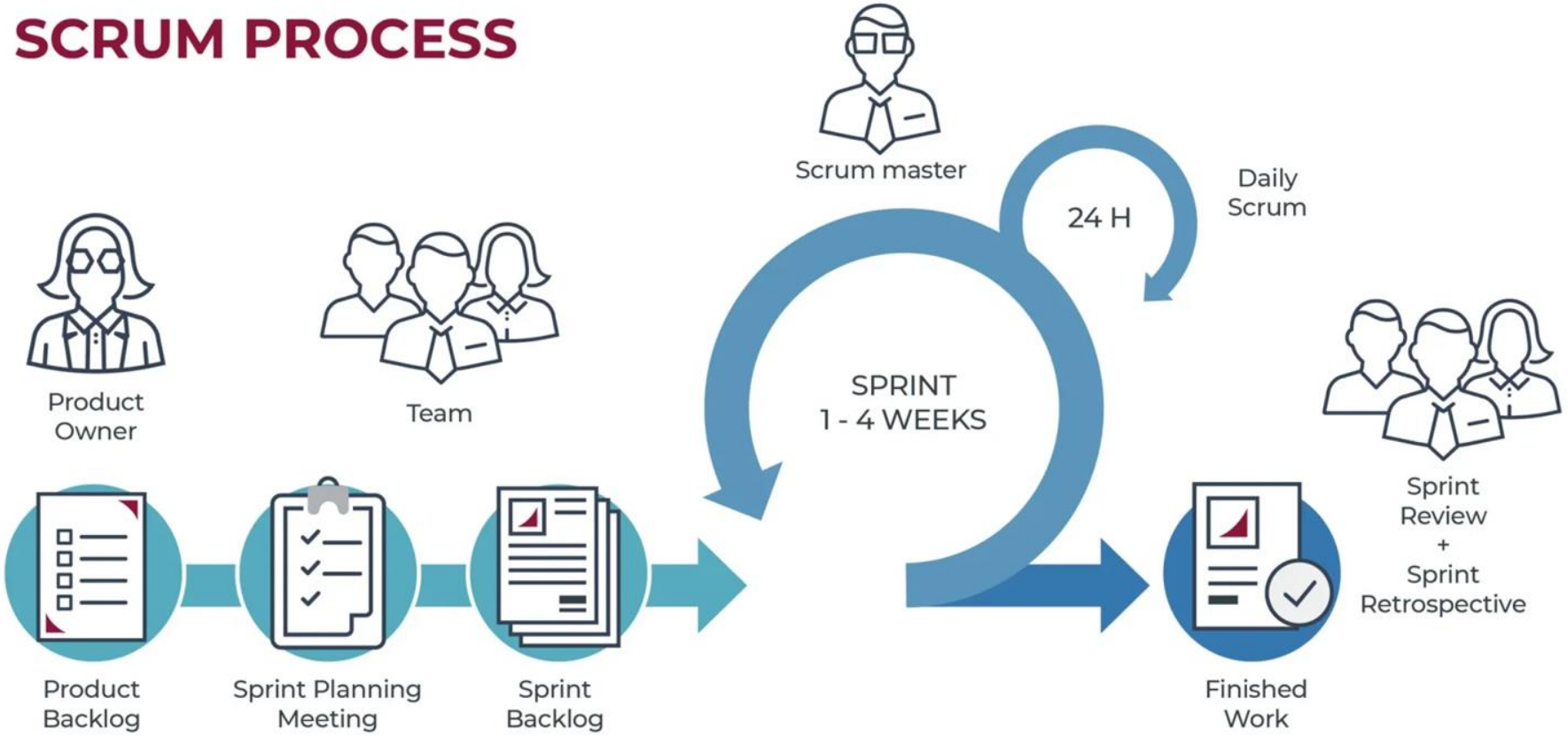
¿Cómo funciona Scrum?

Scrum divide los proyectos en **tareas pequeñas** que se completan en un periodo de tiempo llamado **sprint**, los cuales duran normalmente entre 2 y 4 semanas. En cada sprint los miembros **trabajan en equipo** para llegar a la meta, poniendo en común las **novedades** y problemas en una reunión diaria llamada **Scrum Daily**. Las reuniones se realizan de pie y no pueden durar más de **15 minutos**.

Cuando un sprint llega a su fin, se celebran 2 reuniones. En la primera, llamada **Sprint Review**, se presentan las **novedades a las partes interesadas** y se recogen sus **comentarios** para seguir mejorando el producto. Más tarde se celebra una reunión de retrospectiva interna conocida como **Sprint Retrospective**, donde los miembros del equipo comentan cómo ha ido el sprint, con qué **problemas** se han encontrado y cómo pueden seguir mejorando.

El ciclo vuelve a repetirse una y otra vez, continuando con las tareas pendientes. Este enfoque permite a los equipos ser más flexibles para adaptarse a la volatilidad de los requisitos que aparecen a medida que un proyecto avanza. Todo ello contribuye a una mejora constante del proceso y los resultados.

SCRUM PROCESS



Metodologías y lenguajes

El proceso de desarrollo de software requiere de la aplicación de una **metodología** y algunas **herramientas** consistentes con esa metodología.

Una metodología está formada por un conjunto de **métodos**, **técnicas** y **estrategias**.

Actualmente la metodología más difundida está integrada al **paradigma de programación orientada a objetos**.

Las herramientas más importantes dentro del proceso de desarrollo de software son el **lenguaje de modelado** y el **lenguaje de programación**.

El lenguaje de modelado comienza a utilizarse durante el desarrollo de los requerimientos y permite elaborar diferentes tipos de diagramas.

El lenguaje de programación afecta principalmente a la etapa de implementación.

Lenguaje de modelado y de programación

Un **lenguaje de modelado** es una notación que permite especificar las partes esenciales de un sistema de software.

El lenguaje de modelado más utilizado tanto en el ámbito académico como comercial es **UML**.

Un **lenguaje de programación** es una notación formal con una sintaxis estricta y una semántica rigurosa, que puede ser interpretado y ejecutado por una computadora.

Dos conceptos centrales tanto en la metodología como en las herramientas son objeto y clase.

Calidad

La calidad se evalúa a partir de diferentes factores:

- **Correctitud**: un producto de SW correcto actúa según los requerimientos especificados.
- **Eficiencia**: un producto de SW es eficiente si tiene baja demanda de recursos de hardware, en particular: tiempo de CPU, espacio de memoria y ancho de banda.
- **Portabilidad**: un producto de SW es portable si puede ejecutarse sobre diferentes plataformas de hardware y software.
- **Simplicidad**: Un producto de SW es si es fácil de usar, si su interfaz es amigable y no requiere demasiado entrenamiento o capacitación por parte del usuario.
- **Robustez**: un producto de SW es robusto si reacciona adecuadamente aún en circunstancias imprevistas
- **Usabilidad**: un producto de SW es usable si está disponible en el momento que el usuario lo necesita y el rendimiento está dentro de parámetros establecidos