# Tema 2: El Ciclo de Vida del Software

Introducción a la Ingeniería del Software y los Sistemas de Información I

Ingeniería Informática – Tecnologías Informáticas

Departamento de Lenguajes y Sistemas Informáticos

# Índice

# Objetivos

El objetivo de este capítulo es **introducir los métodos ágiles de desarrollo y gestión de software:**

- Comprender los **fundamentos** de los métodos ágiles de desarrollo y gestión de software.
- Comprender las **implicaciones** de seguir los principios y valores del *manifiesto ágil*.
- Comprender las **diferencias** entre una gestión “tradicional” y una gestión “ágil” del desarrollo.
- Conocer **prácticas de desarrollo ágil**
- Comprender el marco de trabajo de **Scrum** para la gestión ágil de proyectos.

# Índice

# Concepto de Ciclo de Vida

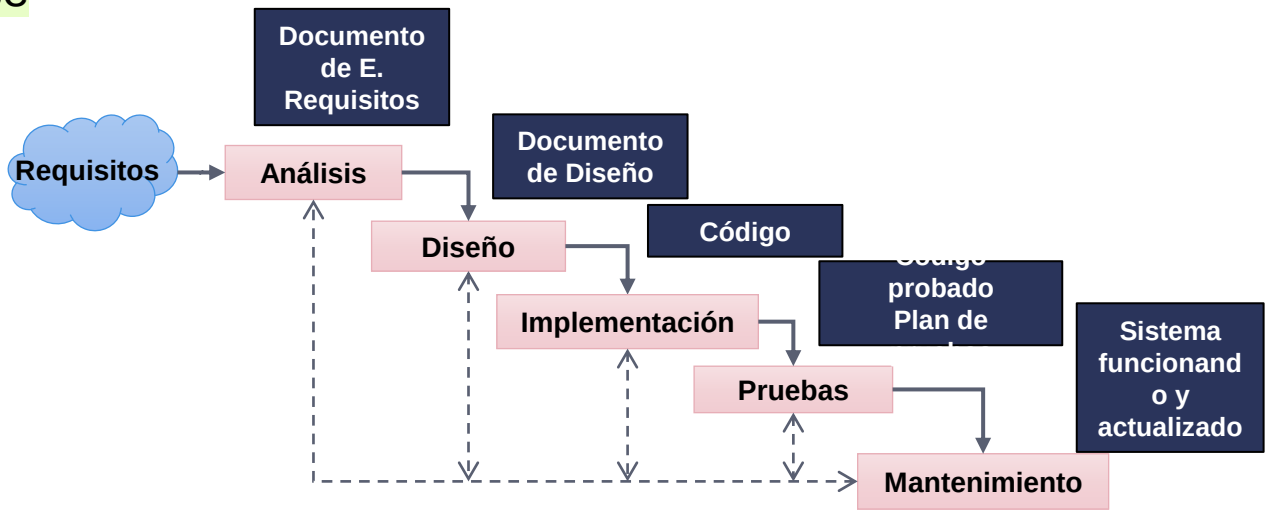
*Un marco de referencia que contiene los **procesos**, las **actividades** y las **tareas** involucradas en el desarrollo, la operación y el mantenimiento de un **producto** software, abarcando la vida del sistema desde su definición hasta su retirada (ISO 12207).*

El ciclo de vida de un proyecto especifica el enfoque general del desarrollo, indicando los **procesos**, **actividades** y **tareas** que se van a realizar y en qué orden, y los **productos** que se van a generar, los que se van a entregar al cliente y en qué orden se van a entregar.

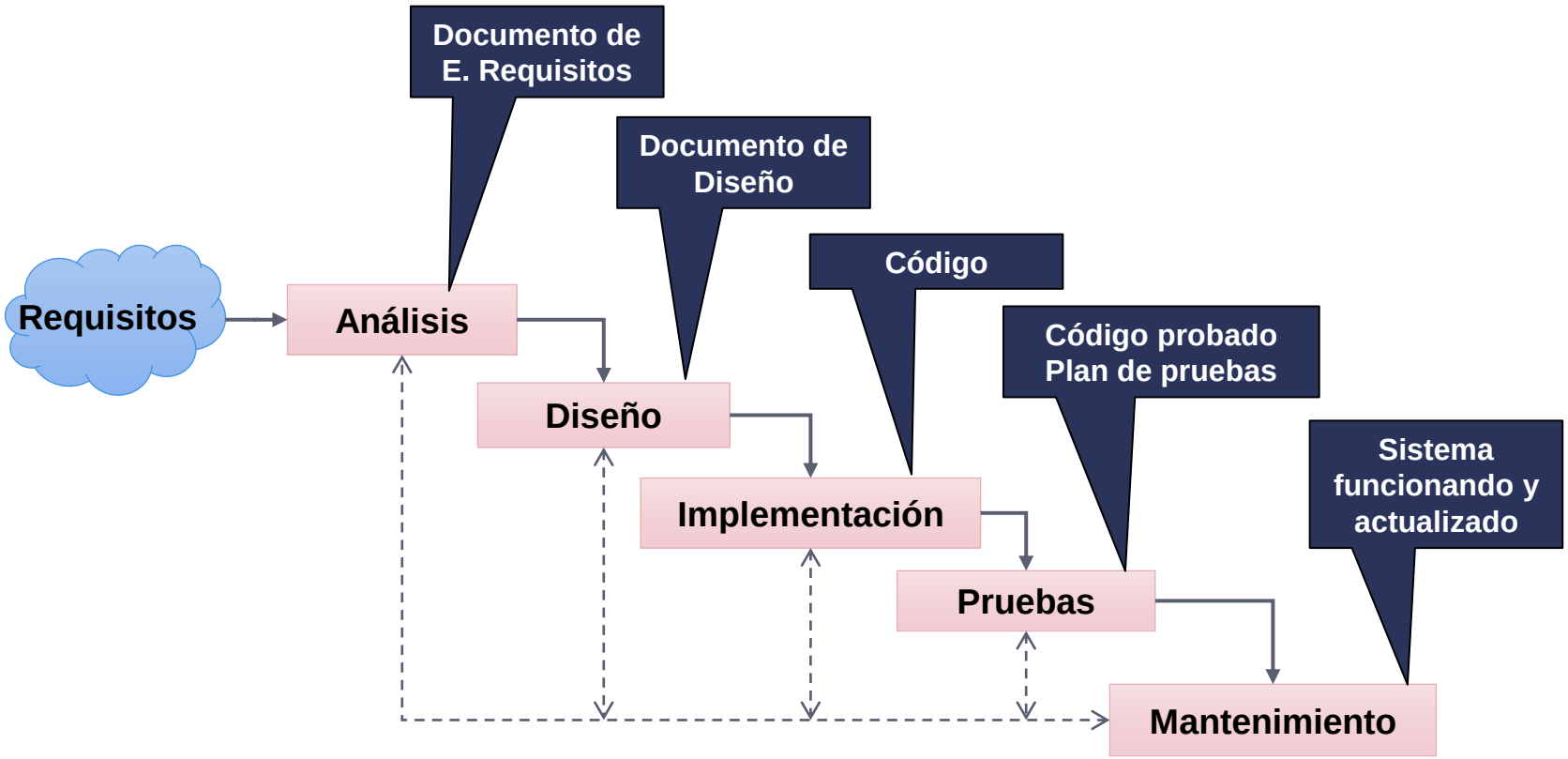
# Métodos de desarrollo “tradicionales” vs “ágiles”

## Métodos “tradicionales” de desarrollo software:

- Modelo de desarrollo secuencial (en cascada) con etapas definidas
- Los resultados de una etapa se utilizan como base para planificar la siguiente actividad del proceso.
- Procesos de desarrollo software basados en una planificación de actividades detallada.
- Especifican completamente los requisitos y luego se diseña, construye y prueba el sistema.
- No están orientados al desarrollo rápido de software.
- A medida que los requisitos cambian tienen que ser reelaborados y probados de nuevo.
- Suelen ser procesos largos, el software final se entrega al cliente mucho después de lo especificado originalmente.

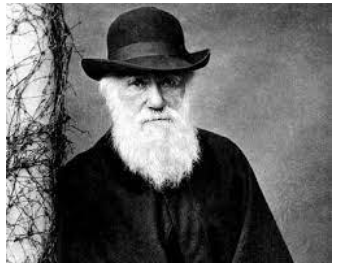


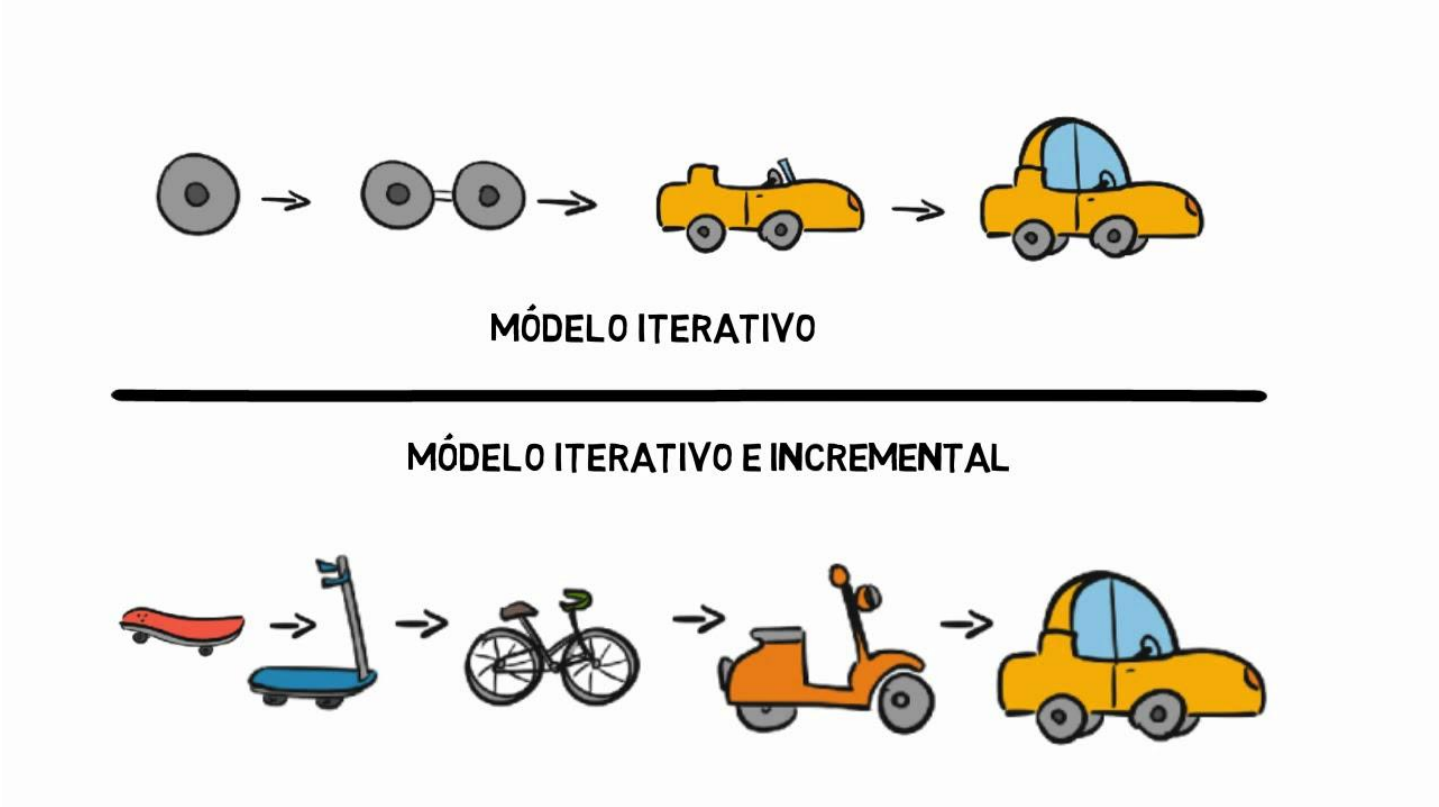
# Ciclo de Vida Clásico



# Métodos de desarrollo “tradicionales” vs “ágiles”

"No es el más fuerte de las **especies** el que **sobrevive**, tampoco es el más inteligente el que **sobrevive**. ***Es aquel que es más adaptable al cambio***“ (Charles Darwin)

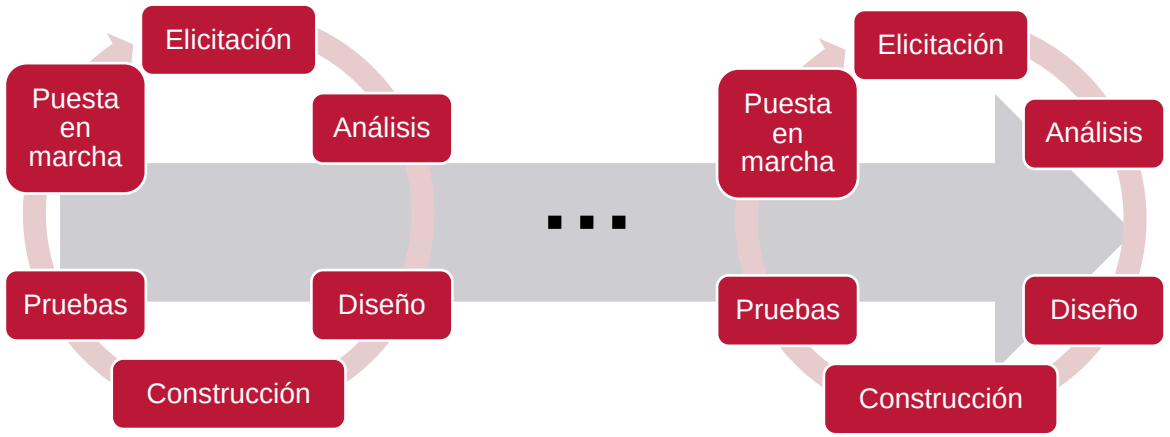




# Métodos de desarrollo “tradicionales” vs “ágiles”

## Métodos “ágiles” de desarrollo software:

- Modelos de desarrollo iterativos e incrementales, con entregas frecuentes
- La filosofía de los métodos ágiles se refleja en el **manifiesto ágil** que acordaron los principales desarrolladores de estos métodos.
- Métodos diseñados para producir **software de valor de forma «flexible» y «adaptable».**
- Los usuarios finales y otras partes interesadas en el sistema participan en la especificación y evaluación de cada incremento.
- Pueden proponerse cambios en el software, así como nuevos requisitos que inicialmente no estaban previstos.
- El proceso de desarrollo es soportado por herramientas para automatizar pruebas, integración y despliegue de código, etc.



# Métodos de desarrollo “tradicionales” vs “ágiles”

| Cuestiones clave | Métodos ágiles | Métodos tradicionales |
|---|---|---|
| **Experiencia del equipo de desarrollo** | Altos niveles de habilidad (experiencia y formación en el negocio y la tecnología). | |
| **Organización del equipo de desarrollo** | Requiere de un equipo auto-organizado | Si parte del desarrollo se subcontrata, es posible que se tengan que elaborar documentos de diseño para comunicarlos entre los equipos de desarrollo. |
| **Tecnologías disponibles para apoyar el desarrollo** | Herramientas de monitorización continua (análisis y evaluación del estado del sistema que se está desarrollando). | |
| **Tamaño del sistema** | Más eficaces con equipos pequeños | Equipos de desarrollo de gran tamaño |
| **Tipo de sistema** | En entornos empresariales donde el software evoluciona rápidamente | Sistemas donde es esencial un análisis completo del sistema (por ej. sistemas de control críticos para la seguridad). |
| **Ciclo de vida útil prevista del sistema** | | Sistemas de larga duración que requieran de más documentación de diseño para comunicar las intenciones originales de los desarrolladores al equipo de soporte. |
| **Regulaciones externas** | | Sistemas que tienen que ser aprobados por un regulador externo (por ejemplo, porque sea un sistema crítico para el funcionamiento de un avión) |

# Manifiesto ágil

**Valores del manifiesto ágil**:

- Individuos e interacciones sobre procesos y herramientas
- Software funcionando sobre documentación extensiva
- Colaboración con el cliente sobre negociación contractual
- Respuesta ante el cambio sobre seguir un plan

# Métodos de desarrollo “tradicionales” vs “ágiles”

**Principios del manifiesto ágil**:

1. Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor.
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.
4. Los responsables de negocio y los desarrolladores trabajamos juntos de forma cotidiana durante todo el proyecto.
5. Los proyectos se desarrollan en torno 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.
7. El software funcionando es la medida principal de progreso.
8. Los procesos Ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante de forma indefinida.
9. La atención continua a la excelencia técnica y al buen diseño mejora la Agilidad.
10. La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial.
11. Las mejores arquitecturas, requisitos y diseños emergen de equipos auto-organizados.
12. A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo para a continuación ajustar y perfeccionar su comportamiento en consecuencia.

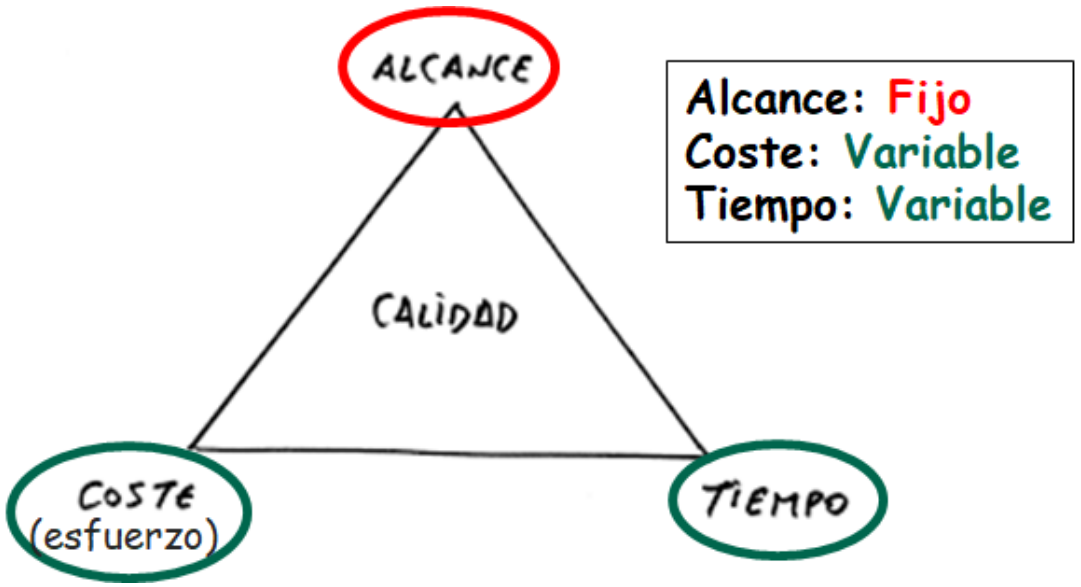
# Índice

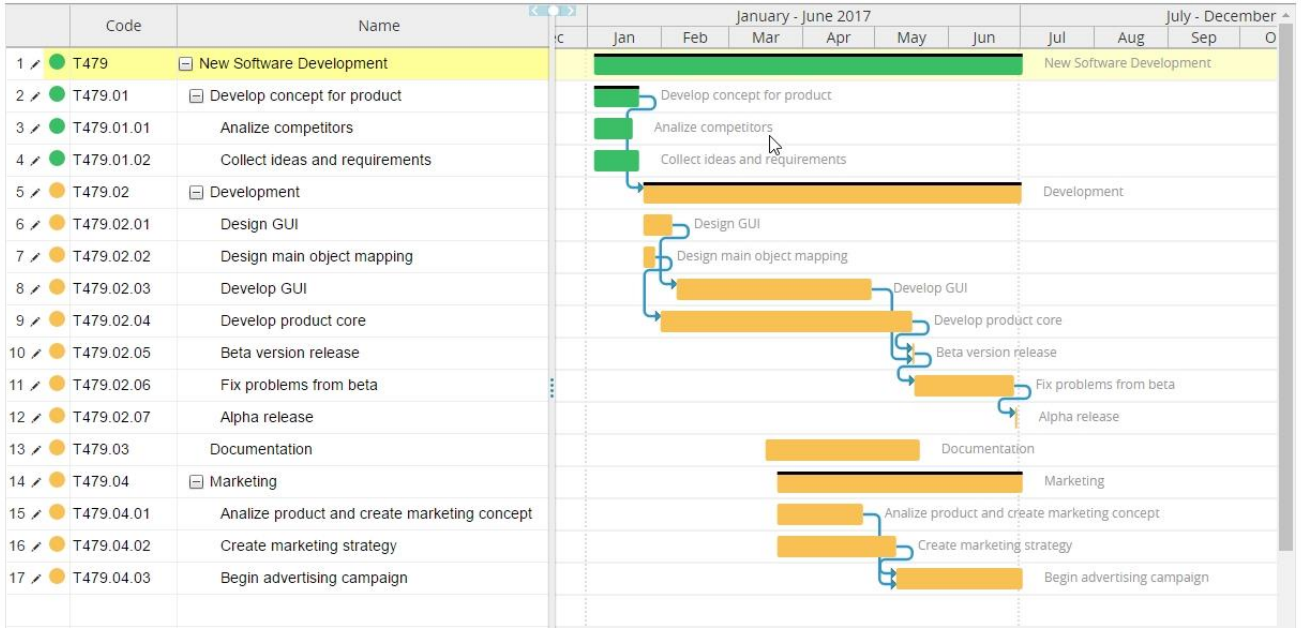
1. Objetivos
2. Métodos de desarrollo “tradicionales” vs “ágiles”
3. **Gestión de proyecto “tradicional” vs “ágil”**
4. Scrum

# Gestión de proyectos “tradicional” vs “ágil”

# Gestión de proyectos “ágiles” vs “tradicionales”

## Gestión de proyectos “tradicional” (plan-driven development) [1]:

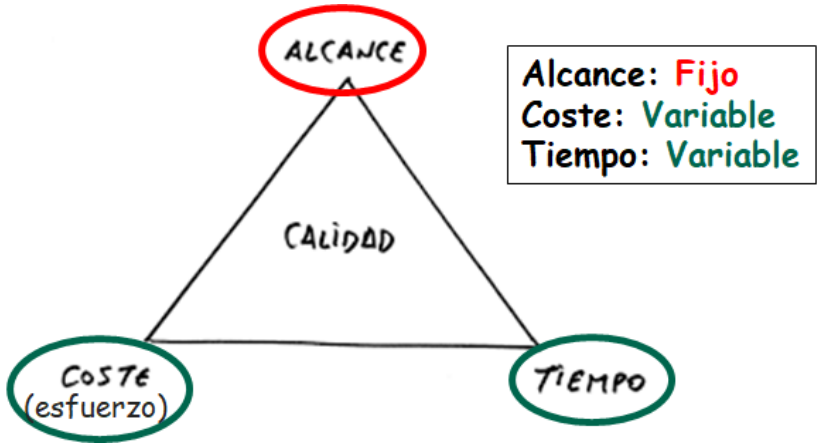


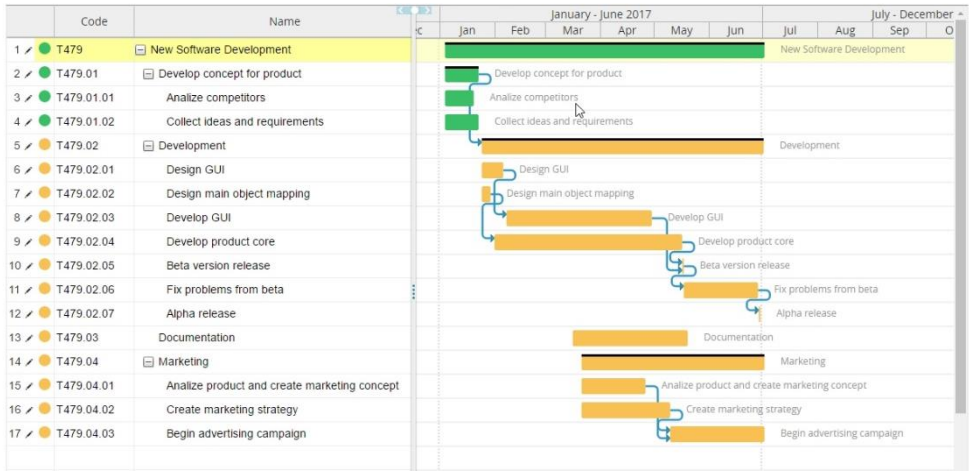




El “alcance” debe obtenerse como resultado al final del proyecto. El “coste” (esfuerzo) y el “tiempo” que dispongamos se va adaptando conforme progresa el proyecto.
**IMPORTANTE**: Debemos asumir la necesidad de dicha adaptación para cumplir con lo inicialmente planificado.

# Gestión de proyectos “ágiles” vs “tradicionales”

## Gestión de proyectos “tradicional” (plan-driven development) [2]:




**PLAN:**

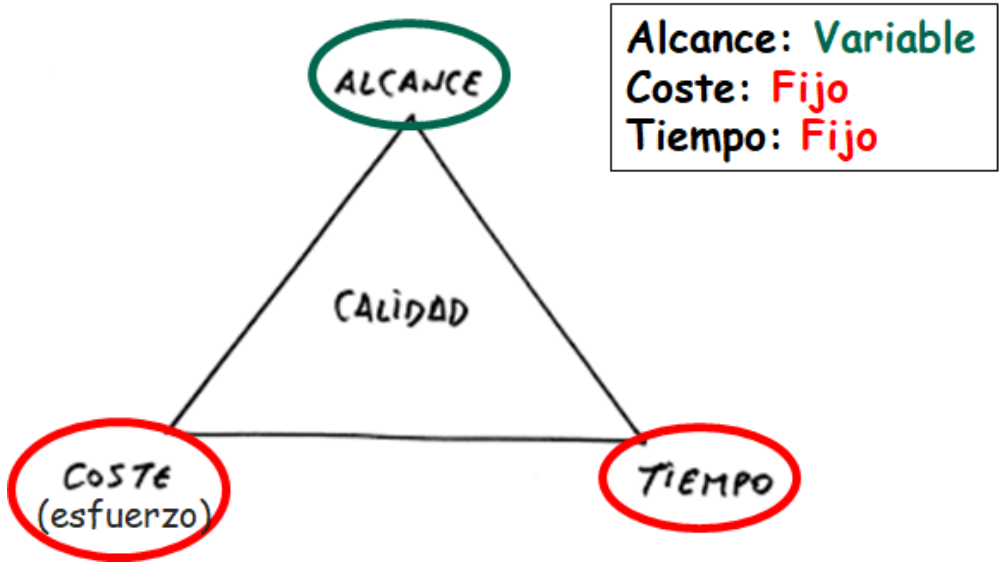
- Se define el alcance.
- Se estima el tiempo (duración de las actividades previstas, por ejemplo, análisis 2 semanas)
- Se estima el coste (previsión y asignación de recursos por actividad, por ejemplo, 2 analistas-programadores)

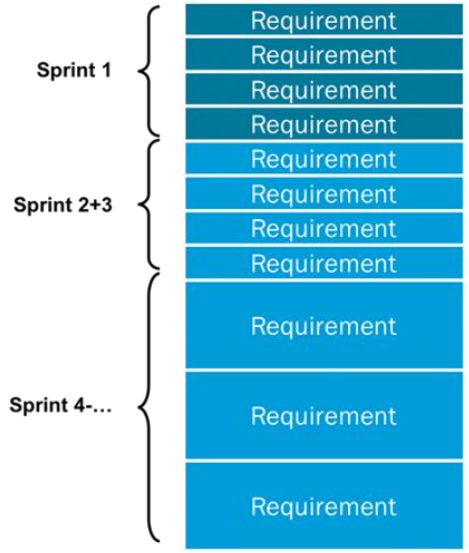
**CONTROL:**

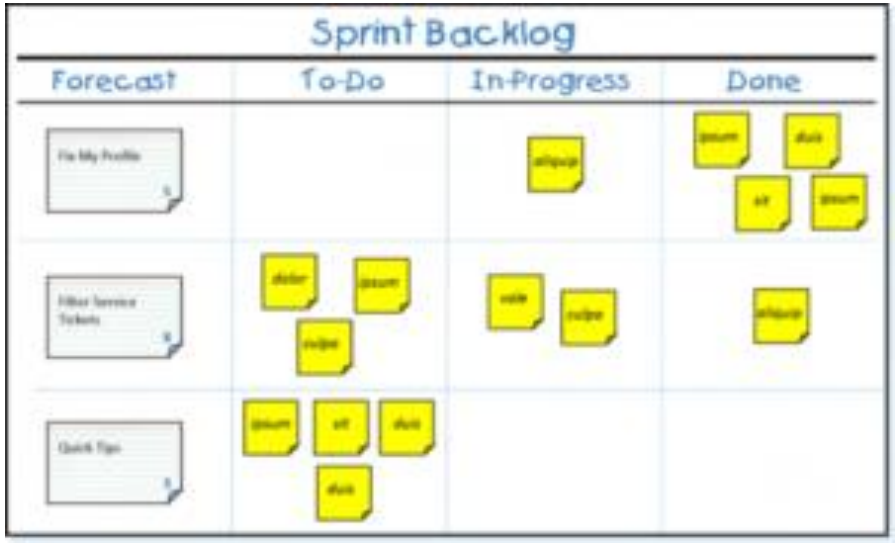
- Se controla el tiempo y el coste. Se realizan las actuaciones necesarias para obtener el alcance previsto. Por ejemplo, si estamos a punto de completar las 2 semanas de análisis, asignamos 1 analista-programador o bien, aumentamos la duración de la actividad 1 semana más.

# Gestión de proyectos “ágiles” vs “tradicionales”

## Gestión de proyectos “ágil” [1]:



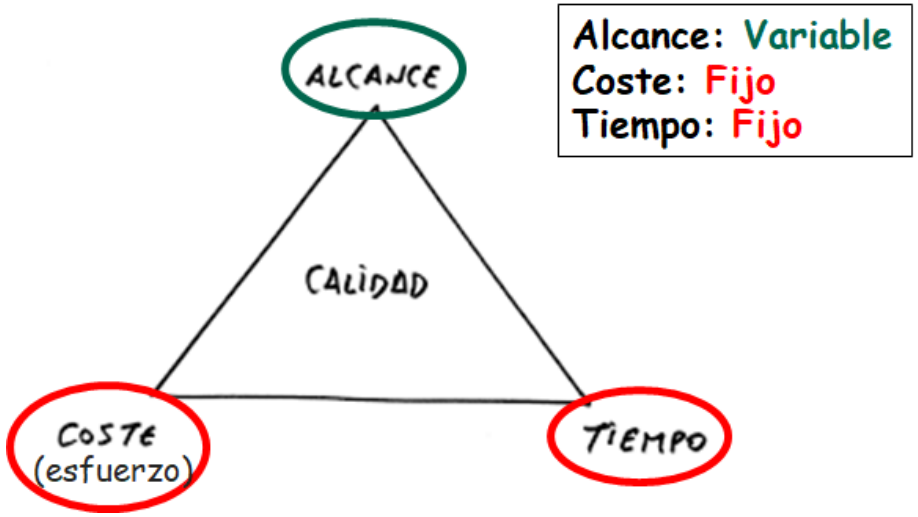



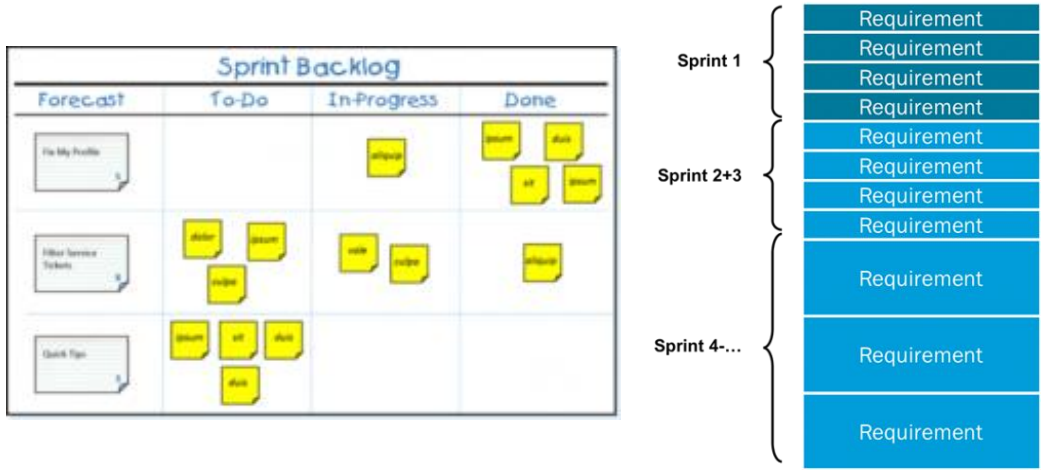




Permiten adaptar el “alcance” al “coste” (esfuerzo) y “tiempo” que dispongamos.
**IMPORTANTE:** Asumimos que no tiene porqué cumplirse el alcance inicial (a menos que se invierta más coste (esfuerzo) y coste que el inicialmente planificado.

# Gestión de proyectos “ágiles” vs “tradicionales”

## Gestión de proyectos “ágil” [2]:




**PLAN**:

- Se estima el alcance. Por ejemplo, listado de requisitos o tareas priorizados en un backlog.
- Se fija la duración de los Sprints y se estima el número necesario. Por ejemplo, sprint de 2 semanas de duración. Se realizarán 10 Sprints.
- Se fija el equipo de desarrollo (implica un consumo del presupuesto fijo por Sprint). Por ejemplo, el equipo de desarrollo lo formarán un equipo de 5 profesionales expertos en UX, análistas-programadores, programadores y testing.

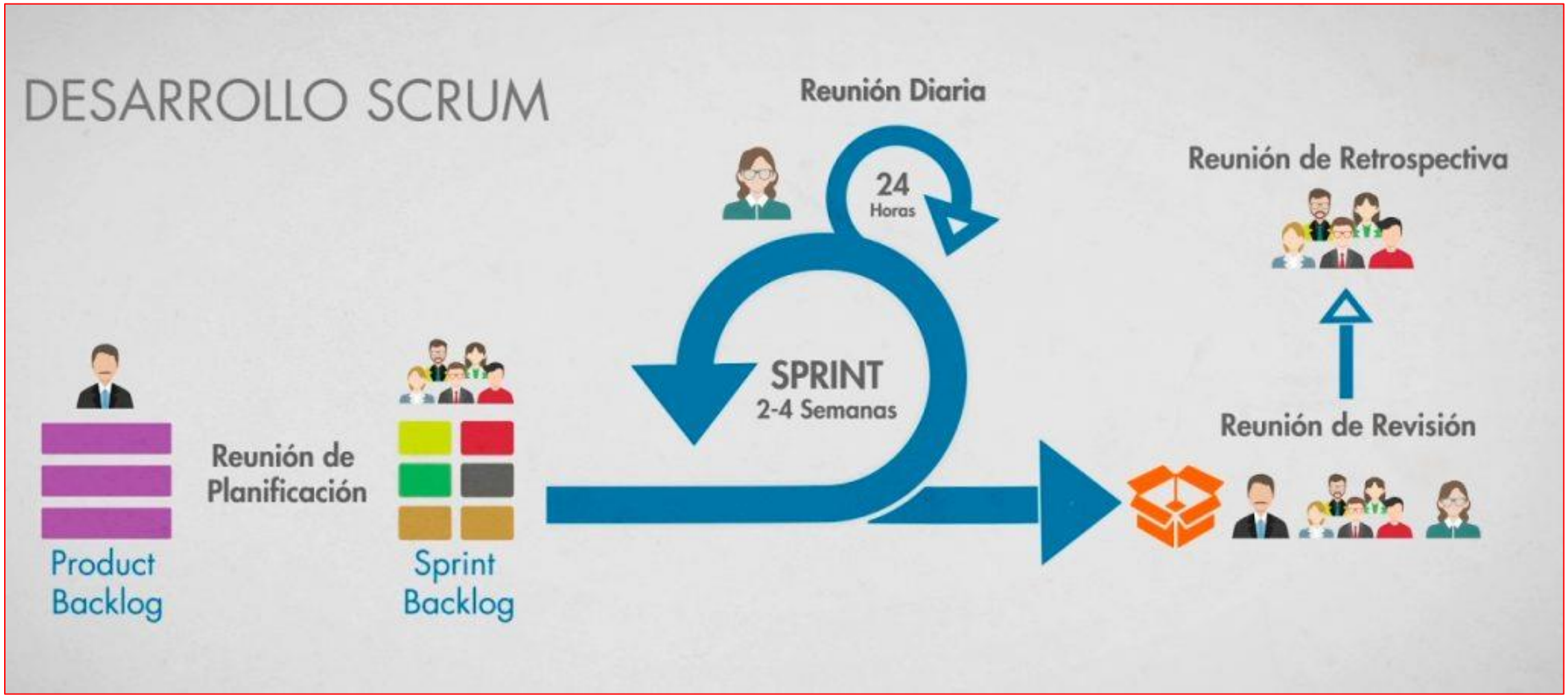
**CONTROL**:

- Se controla el alcance, que irá adaptandose a las circunstancias del momento (cambio de requisitos), presupuesto y a los Sprints restantes. Por ejemplo, es posible introducir nuevos requisitos en cualquier momento, pero no cambiar coste y tiempo.

# Índice

1. Objetivos
2. Métodos de desarrollo “tradicionales” vs “ágiles”
3. Gestión de proyecto “tradicional” vs “ágil”
4. **Scrum**

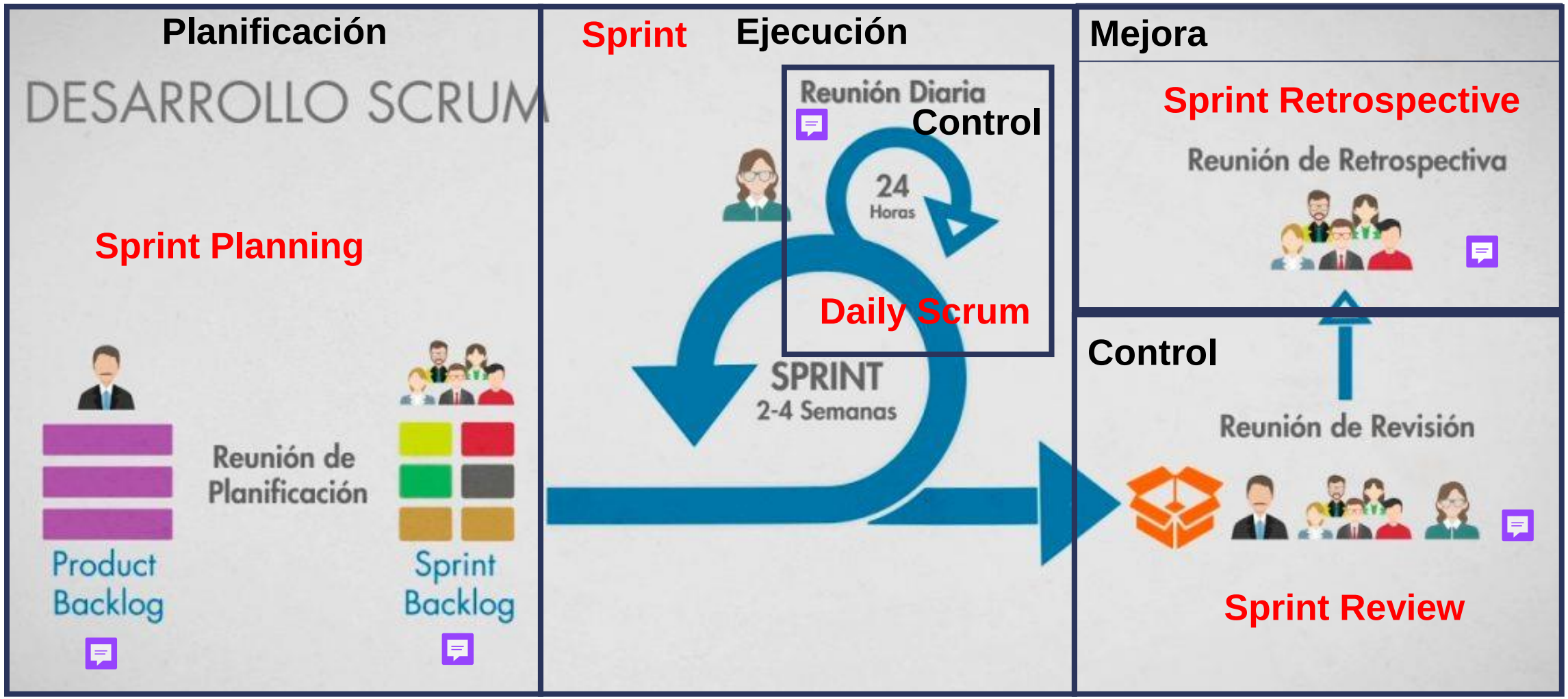
# Scrum




https://scrumguides.org/

# Scrum

## Proceso y actividades en SCRUM



# Scrum

**Planificación del Sprint:**

- Inicia el Sprint y participa el equipo Scrum.
- Aborda tres temas:
  - ¿Por qué este Sprint aporta valor? Cómo ayuda a avanzar hacia el Product Goal.
  - ¿Qué se puede hacer en este sprint? Establecer el Sprint Goal.
  - ¿Cómo se realizará el trabajo elegido? Definir el trabajo para crear un incremento.
- El resultado es el Sprint Backlog compuesto del sprint Goal, los items seleccionados de Product Backlog y el plan para ponerlos en producción.
- Duración máxima 8 horas para un Sprint de 4 semanas.

**Reunión diaria:**

- Permite inspeccionar el logro del objetivo del Sprint y adaptar el Sprint Backlog según necesidad.
- Reunión diaria de desarrolladores (developers), de 15 minutos, a la misma hora y lugar.
- Product Owner y Scrum Master participan si están trabajando en ítems del Sprint Backlog.

# Scrum

**Reunión de revisión:**

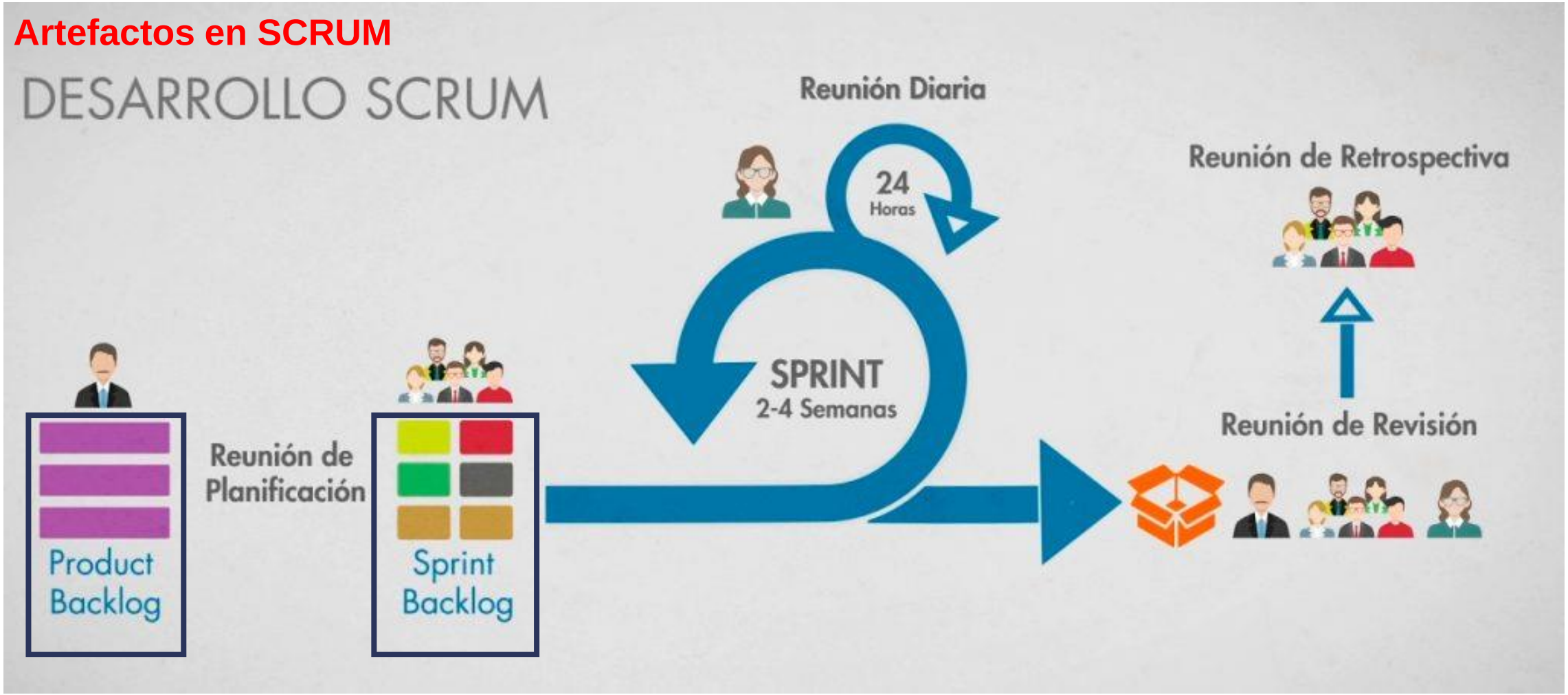
- El objetivo es inspeccionar los resultados (outcomes) del Sprint y determinar adaptaciones futuras.
- Participa equipo Scrum y stakeholders.
- El equipo Scrum presenta los resultados de su trabajo a los stackeholders y se discute el progreso hacia el objetivo del producto.
- Se decide qué hacer a continuación: el Product Backlog puede ajustarse.
- Es una sesión de trabajo, no una presentación.
- Penúltimo evento antes de terminar el Sprint, con una duración máxima 4 horas para un Sprint de 4 semanas.

**Reunión de retrospectiva:**

- Participa el equipo Scrum.
- Duración máxima 3 horas para Sprints de 4 semanas.
- El Equipo Scrum inspecciona cómo fue el último Sprint con respecto a las personas, las interacciones, los procesos, las herramientas y su *Definición de Hecho*.
- Se analiza qué salió bien, qué problemas se encontraron y cómo se resolvieron (o no) y se plantean cambios (acciones de mejora) para el próximo sprint.
- La retrospectiva da por terminado el sprint.

# Scrum

**Artefactos en SCRUM**




**ProductBacklog** **SprintBacklog** **Tablero Scrum (como herramienta de gestión)**

# Scrum

**Product Backlog:**

- Lista dinámica y ordenada de lo que es necesario para mejorar el producto.
- Los ítems del Product Backlog que pueden ser hechos (cumplir la definición de hecho) pueden seleccionarse para un sprint.
- El Product Backlog describe lo necesario para alcanzar el Product Goal.
- Responsable el Product Owner.
- Un producto sólo puede tener un único Product Backlog (y Product Owner).

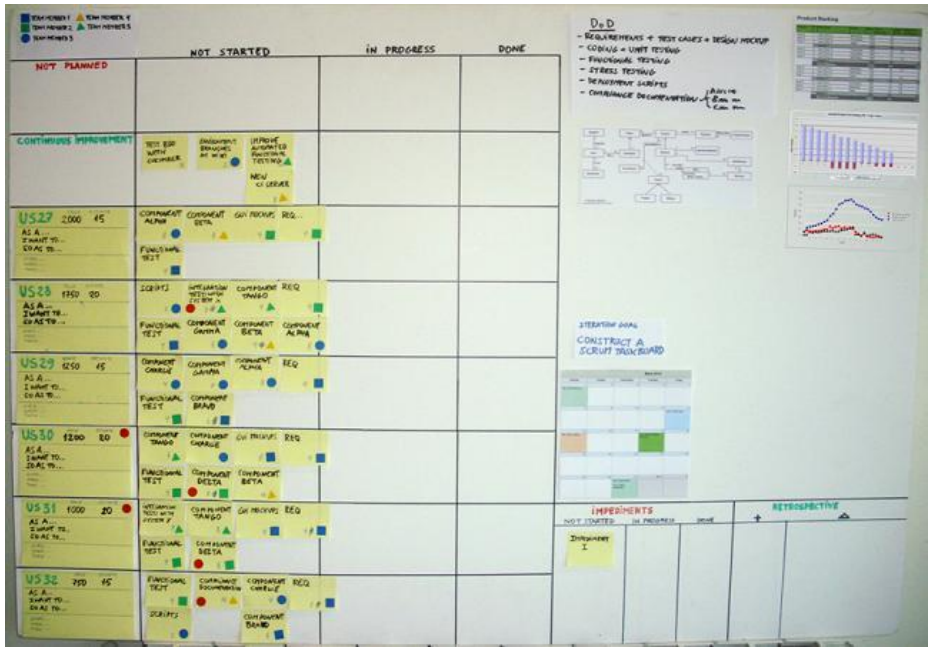
**Sprint Backlog:**

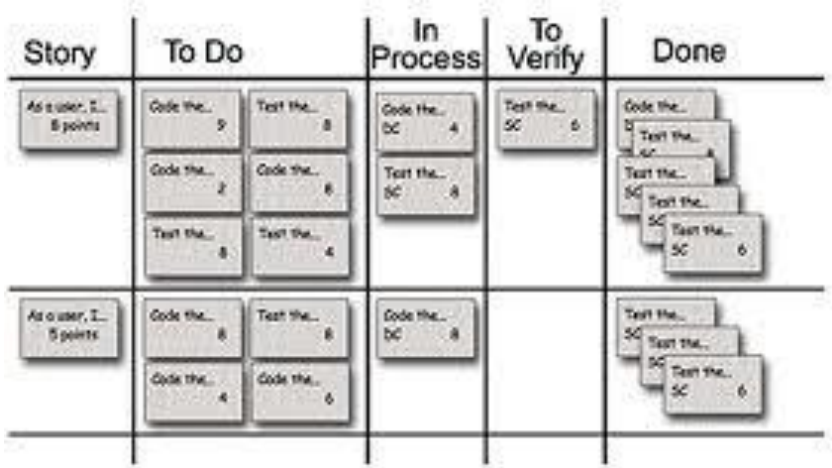
- Objetivo del sprint (compromiso de los desarrolladores)
- Items del Product Backlog seleccionados para el sprint.
- Plan para la entrega del incremento.
- Se mantiene actualizado a lo largo del sprint a medida que se aprende.
- Es la fuente principal de información del Daily Scrum.

# Scrum

## Artefactos en SCRUM

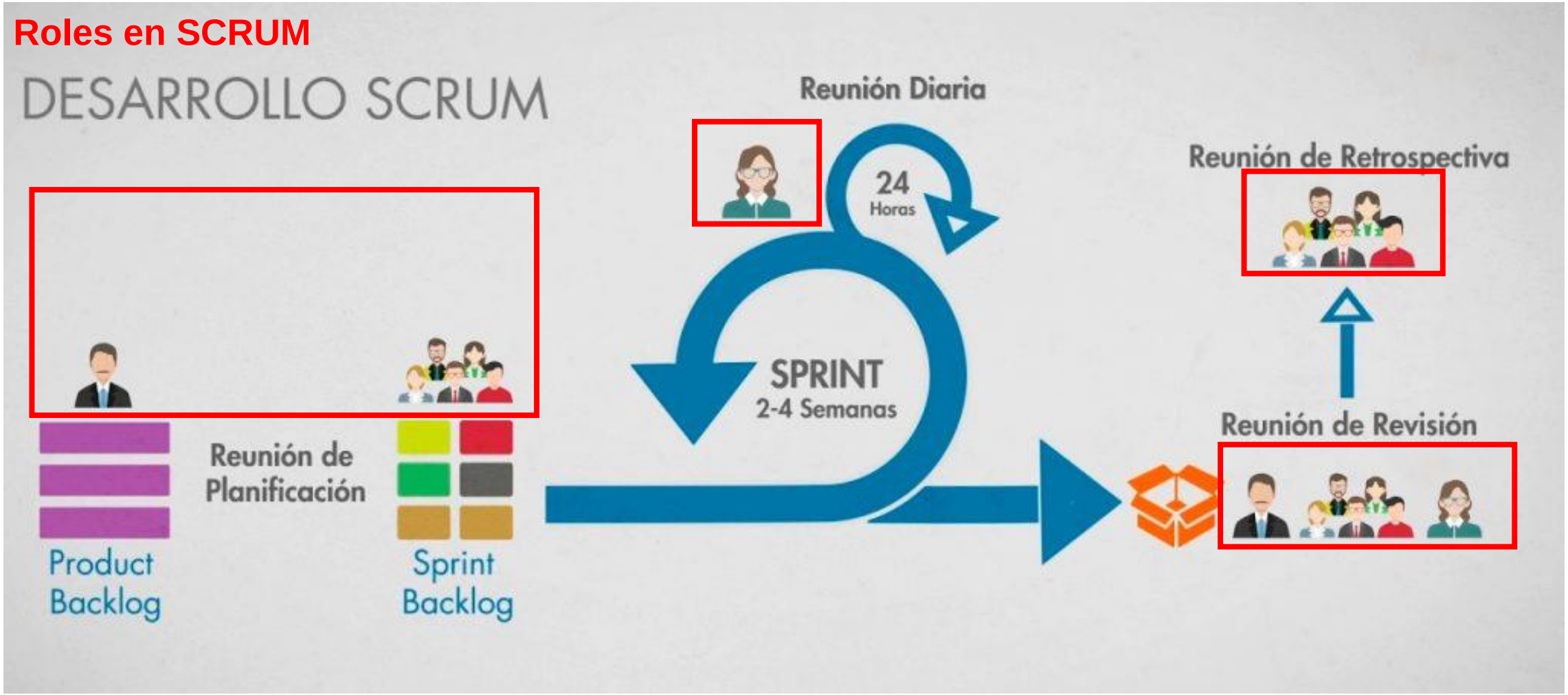
### Tablero SCRUM



# SCRUM

## Roles en SCRUM




**Scrum Master  Product Owner  Developers**

# Scrum

**Scrum Master:**

- Unipersonal.
- Responsable de establecer Scrum según la Guía de Scrum.
- Responsable de la efectividad del Scrum Team.
- Sirve al Product Owner.
- Sirve a la organización.

**Developers:**

- Todas las personas del equipo Scrum comprometidos en la creación de un incremento usable en cada sprint.
- Responsables de Crear el Sprint Backlog, Instituir calidad siguiendo la Definición de Hecho (DoD) y adaptar su plan hacia el Sprint Goal cada día.

**Product Owner:**

- Unipersonal.
- Responsable de maximizar el valor del producto resultante del trabajo del Equipo Scrum.
- Responsable de la gestión del Product Backlog (directamente o por delegación).

# Resumen

- Los **métodos ágiles** son métodos de desarrollo iterativos e incrementales que, cuando nos tenemos que enfrentar a un proyecto donde:
  - Existe incertidumbre en el negocio (qué hay que hacer) y en la tecnología (cómo hacerlo).
  - Existe presión por entregar rápido y ofrecer valor a los interesados del proyecto cuanto antes.
- Nos permiten **"adaptar" el alcance del proyecto (los requisitos) al esfuerzo (coste) y tiempo que dispongamos**. Implican directamente a todos los interesados del proyecto en el proceso de desarrollo.
- La decisión de utilizar un enfoque de desarrollo "ágil" o "tradicional" debe **depender del tipo de software que se desarrolle**, de las **capacidades del equipo de desarrollo** y de la **cultura de la empresa** que desarrolla el sistema.

# Resumen

- Las **prácticas de desarrollo ágil** incluyen desarrollo y gestión de requisitos expresados en forma de *historias de usuario*, *programación por parejas*, *refactorización*, la *integración continua* y un *desarrollo donde las pruebas son prioritarias*.
- **Scrum** proporciona un **marco de trabajo** para la gestión de proyectos ágiles.
  - Se centra en torno a un **conjunto de sprints**, que son **períodos de tiempo fijo** (la duración de cada Sprint debe ser la misma) y **coste fijo** (mismo equipo en cada Sprint), en los que se desarrolla un **incremento del sistema.**
  - La planificación de se basa en la **priorización de un backlog de trabajo** y en la **selección de las tareas más prioritarias** para un sprint (permite ser flexible y adaptar el alcance al presupuesto restante en cualquier momento).

# Tema 2: El Ciclo de Vida del Software

Introducción a la Ingeniería del Software y los Sistemas de Información I

Ingeniería Informática – Tecnologías Informáticas

Departamento de Lenguajes y Sistemas Informáticos