

CURSO DE POSGRADO

# Técnicas y Gestión de las Pruebas de Software

---

Darío Macchi

DOCENTE (invitado)

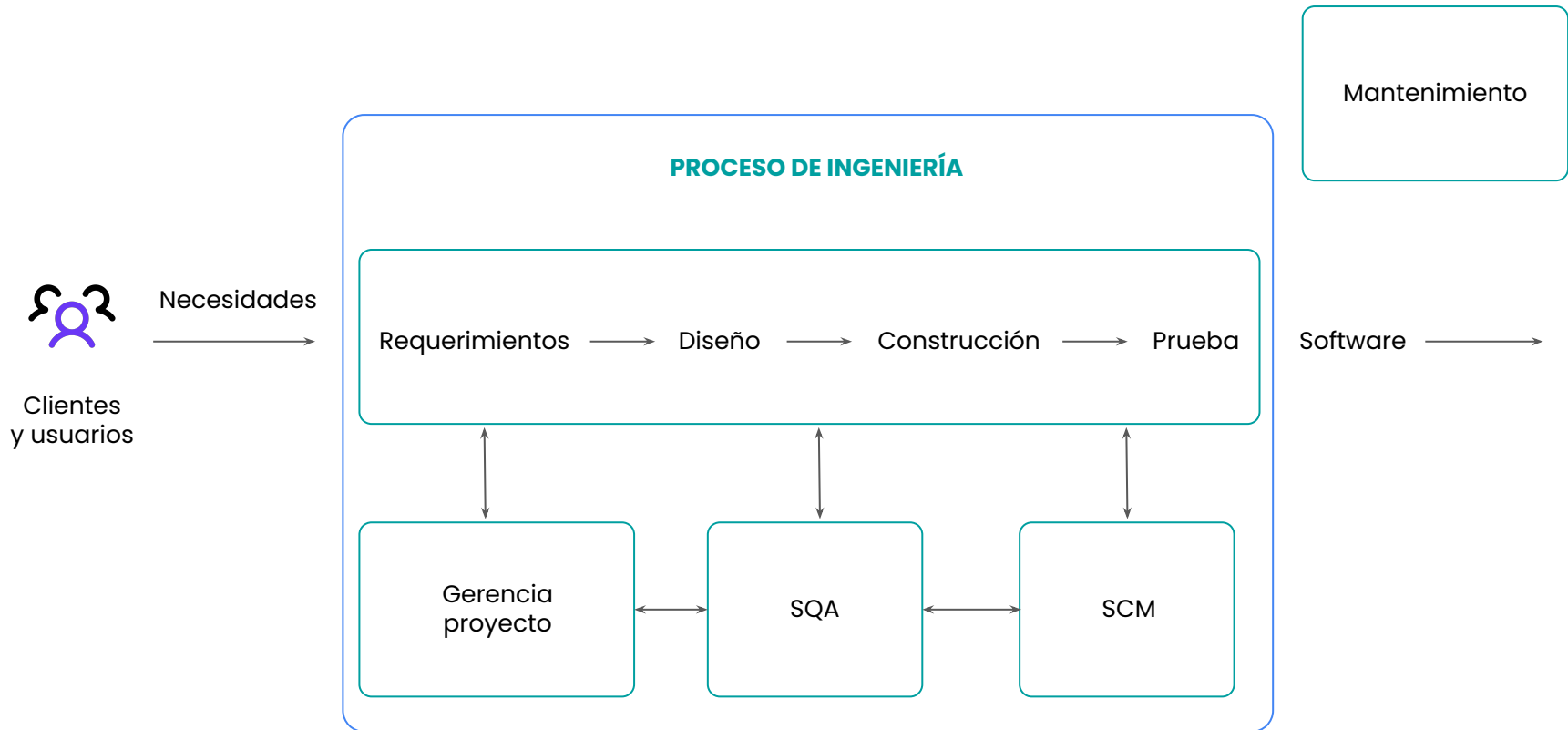
# Procesos de pruebas de software

## Generalidades

# Qué espero de esta clase

- Entender:
  - *testing* como cualquier actividad del desarrollo de SW, sigue un proceso
  - modelo de desarrollo condiciona dicho proceso
  - relación del proceso de pruebas con el proyecto que lo enmarca

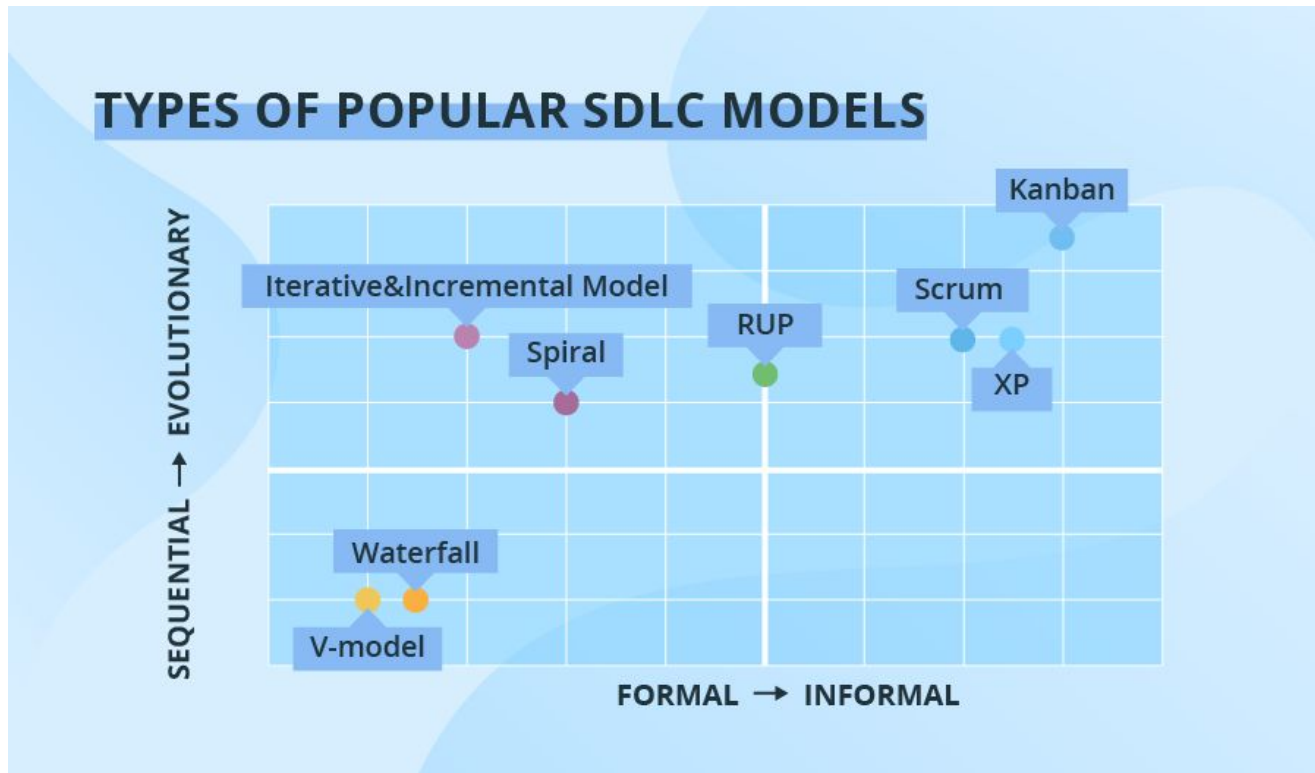
# Proceso de Ingeniería



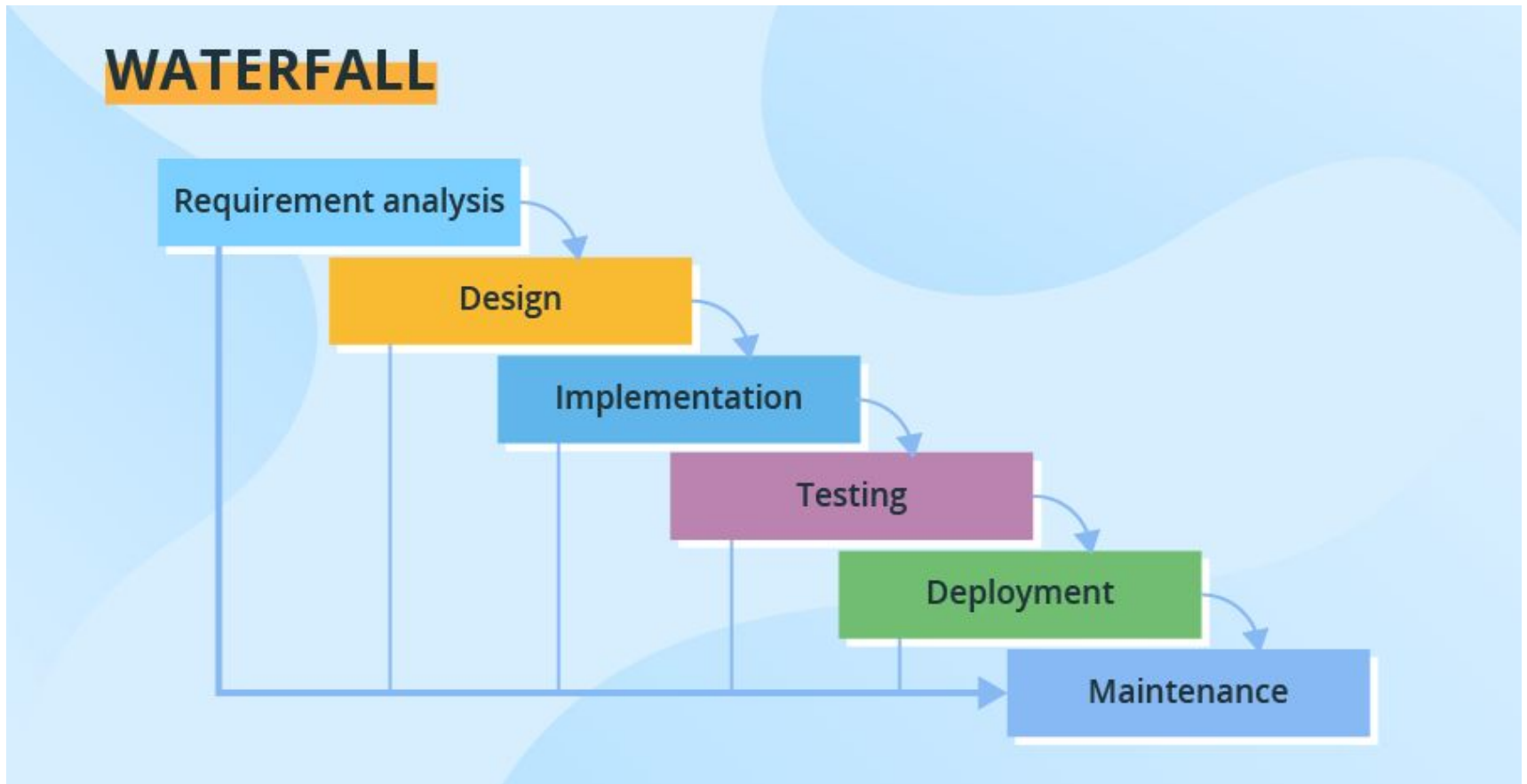
# Modelos de desarrollo de software

# Modelos de desarrollo de software

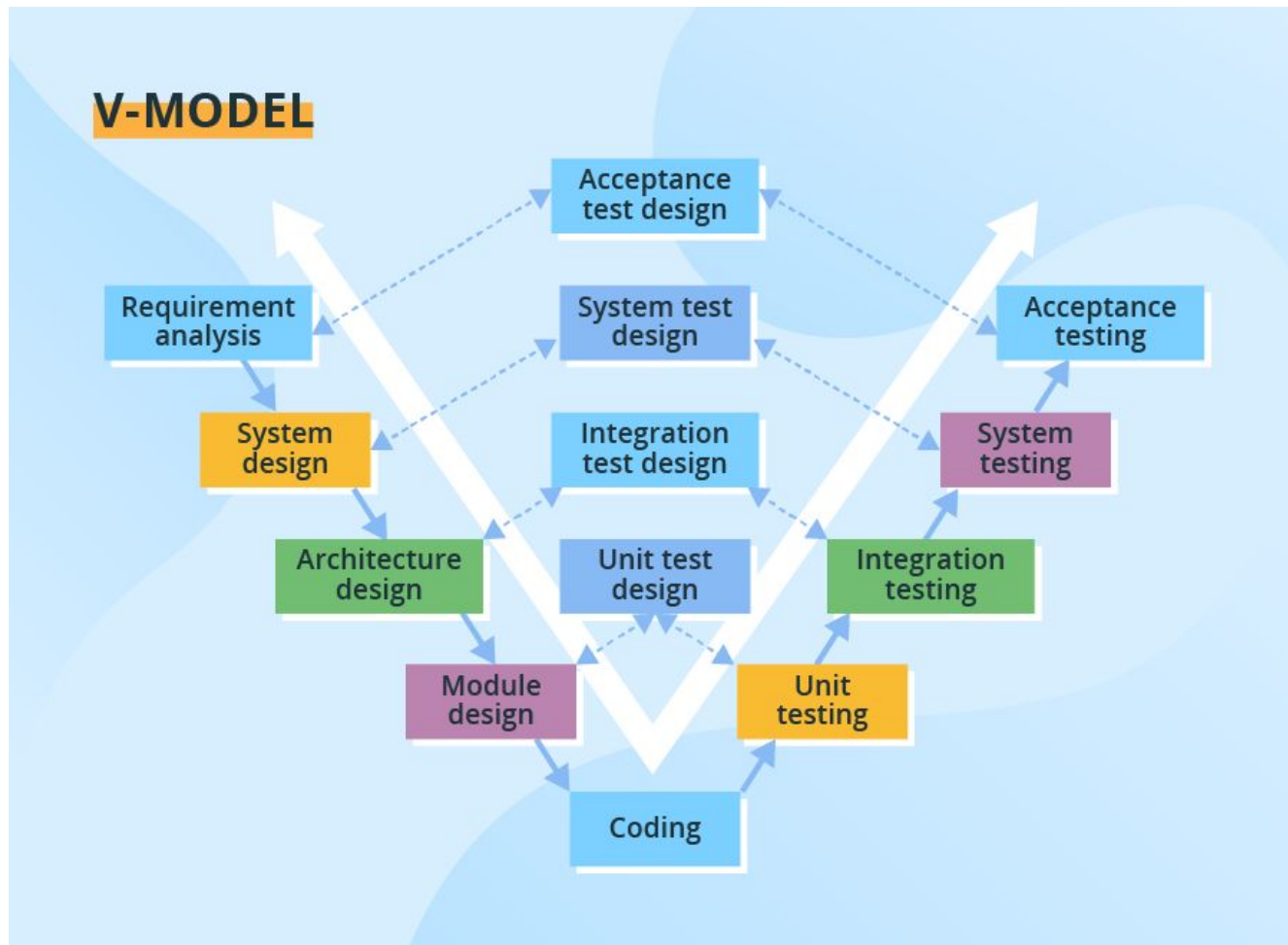
- 50+ modelos reconocidos en uso
- Solo vamos a repasar los populares:



# Cascada

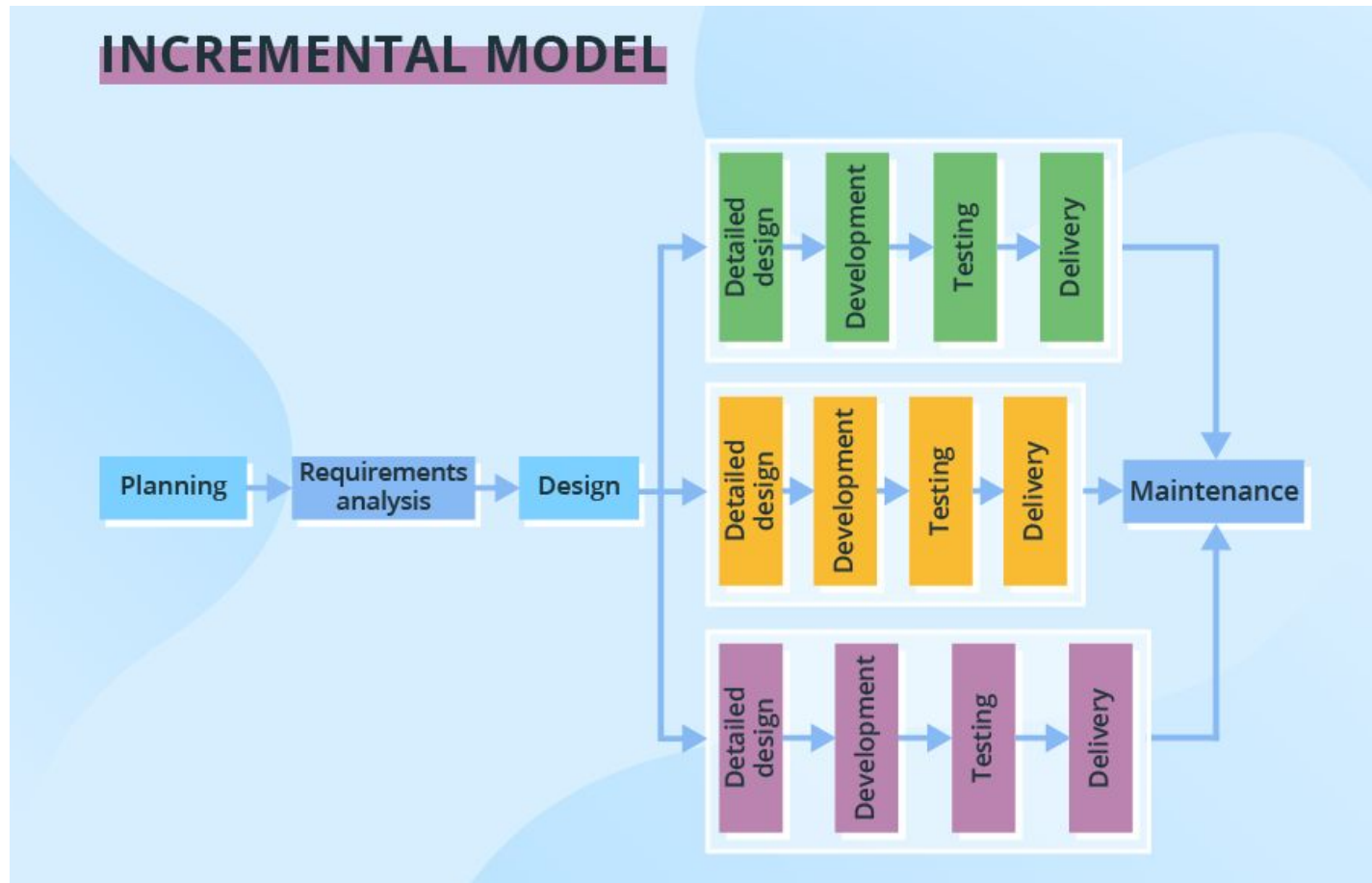


# Modelo en V

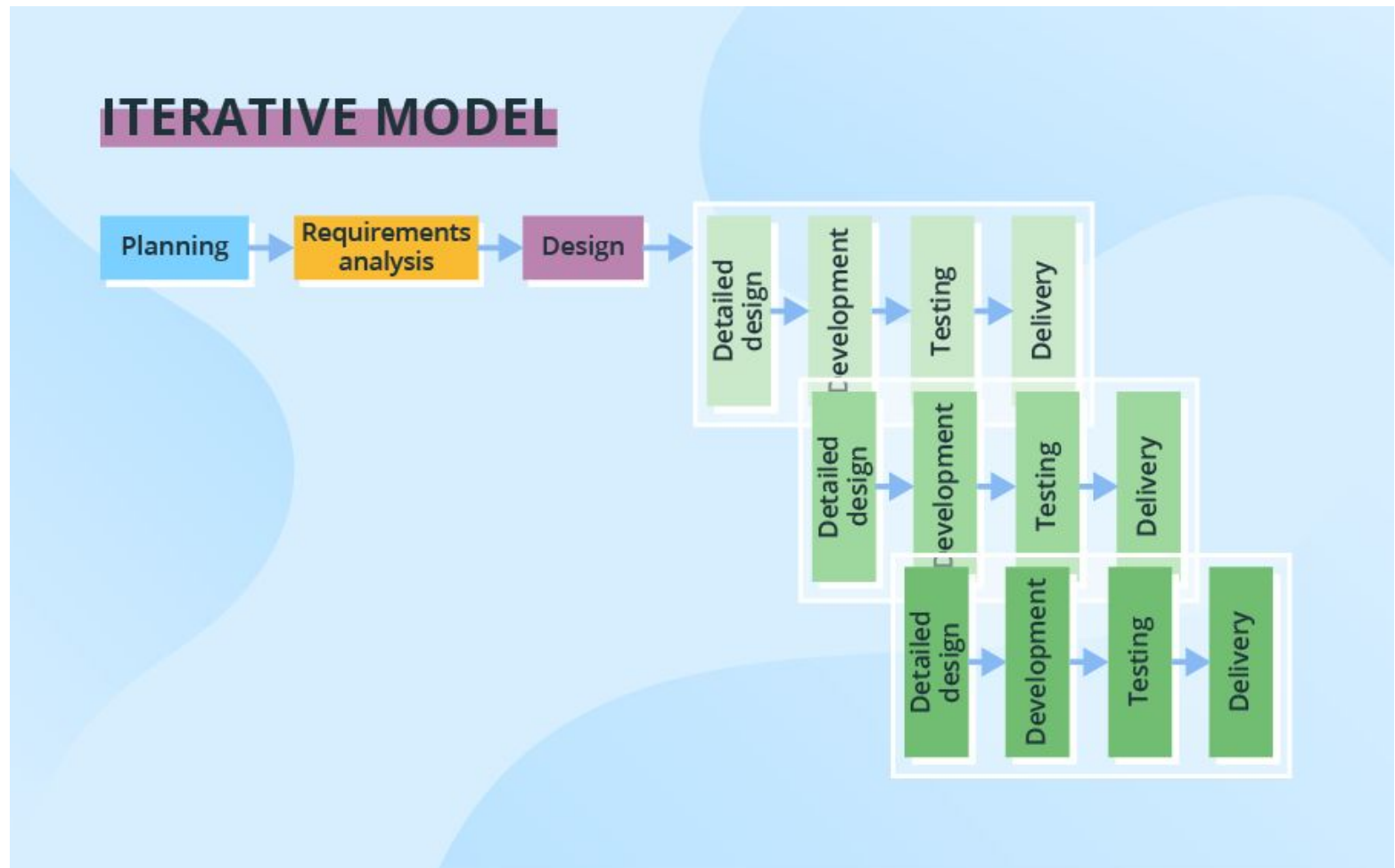




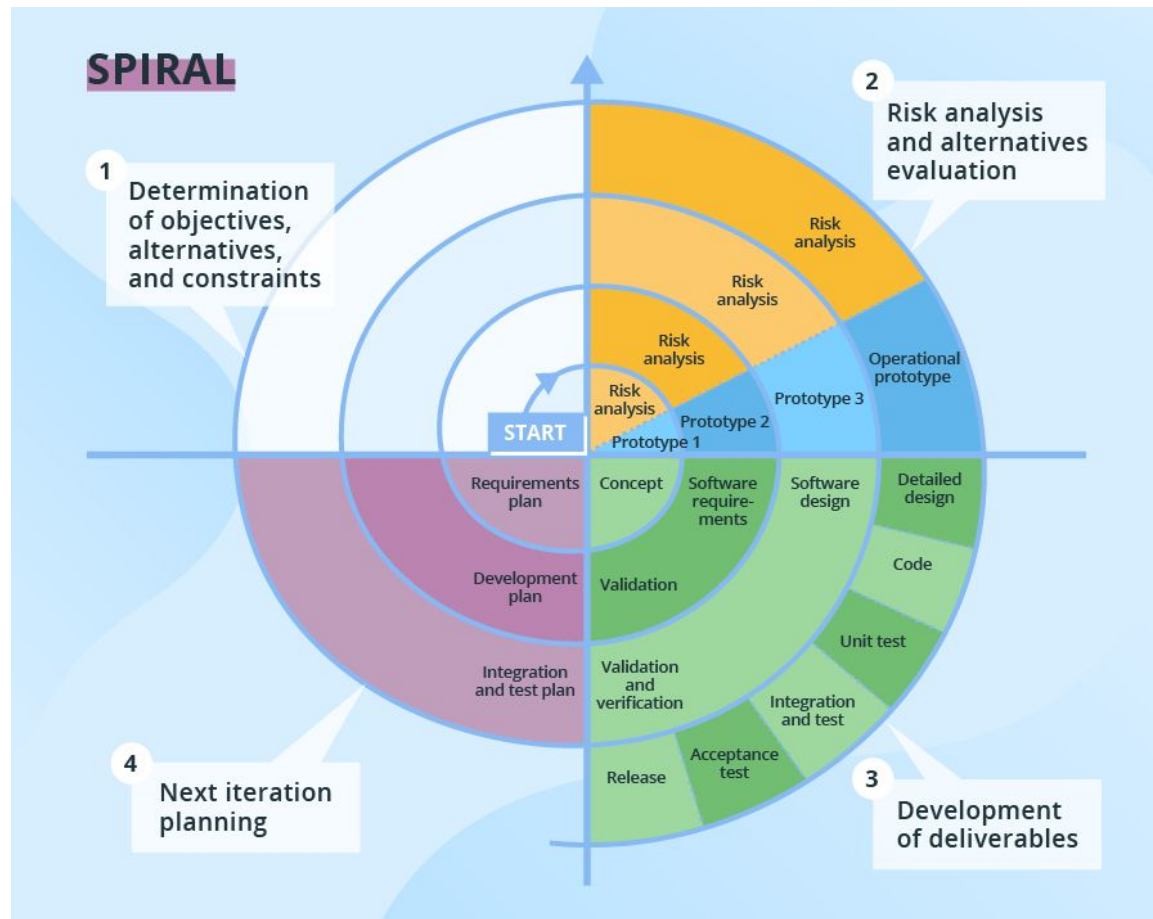
# Modelo incremental



# Modelo iterativo

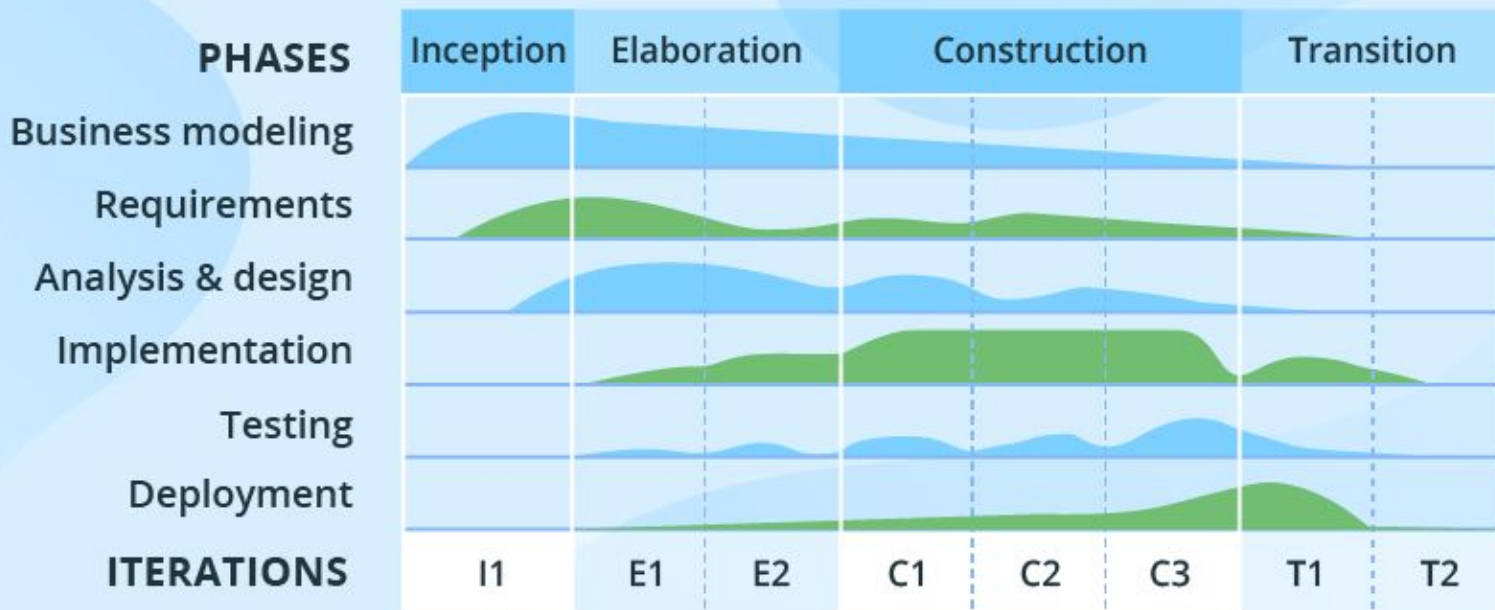


# Modelo en espiral

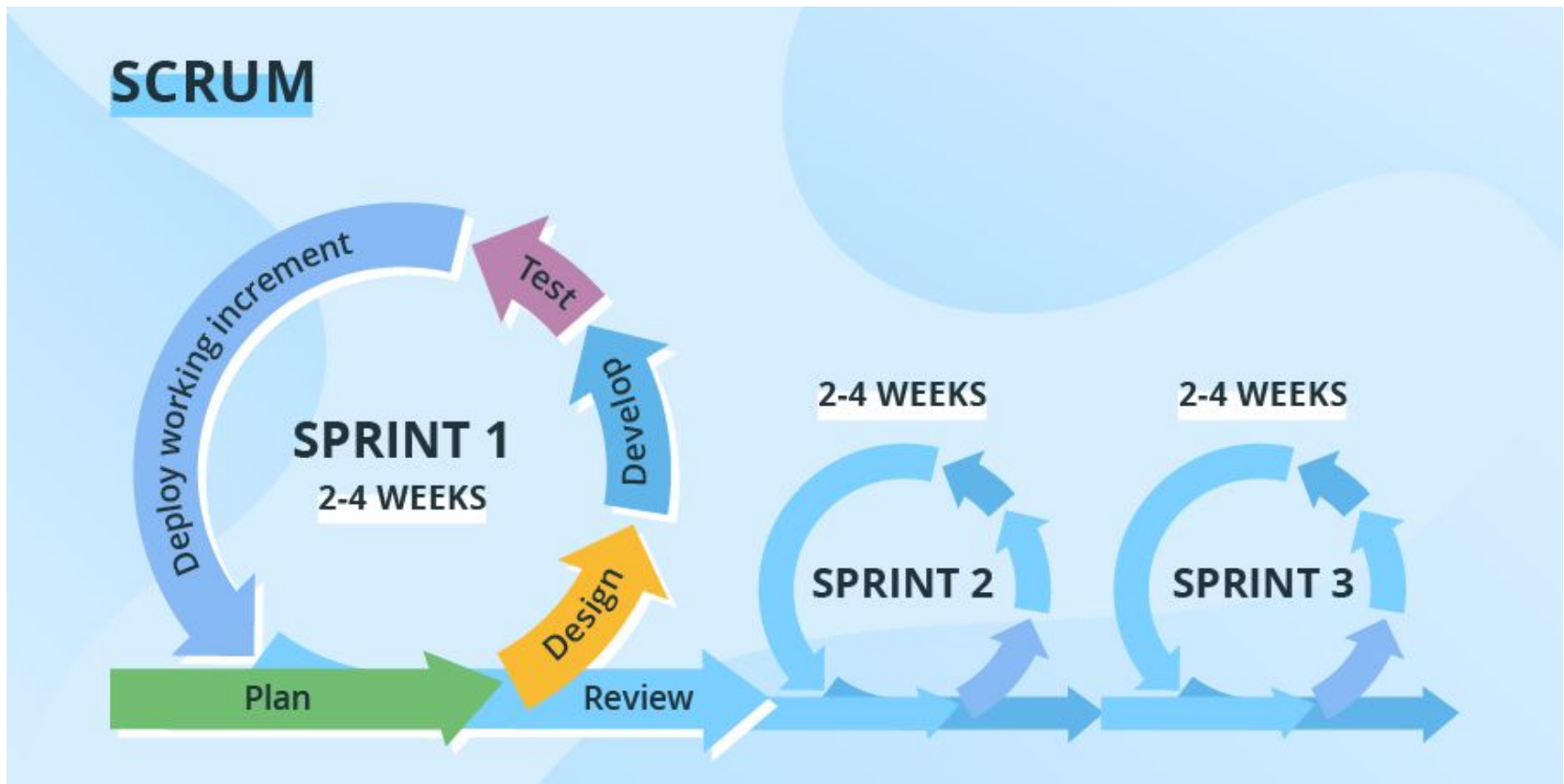


# Modelo RUP

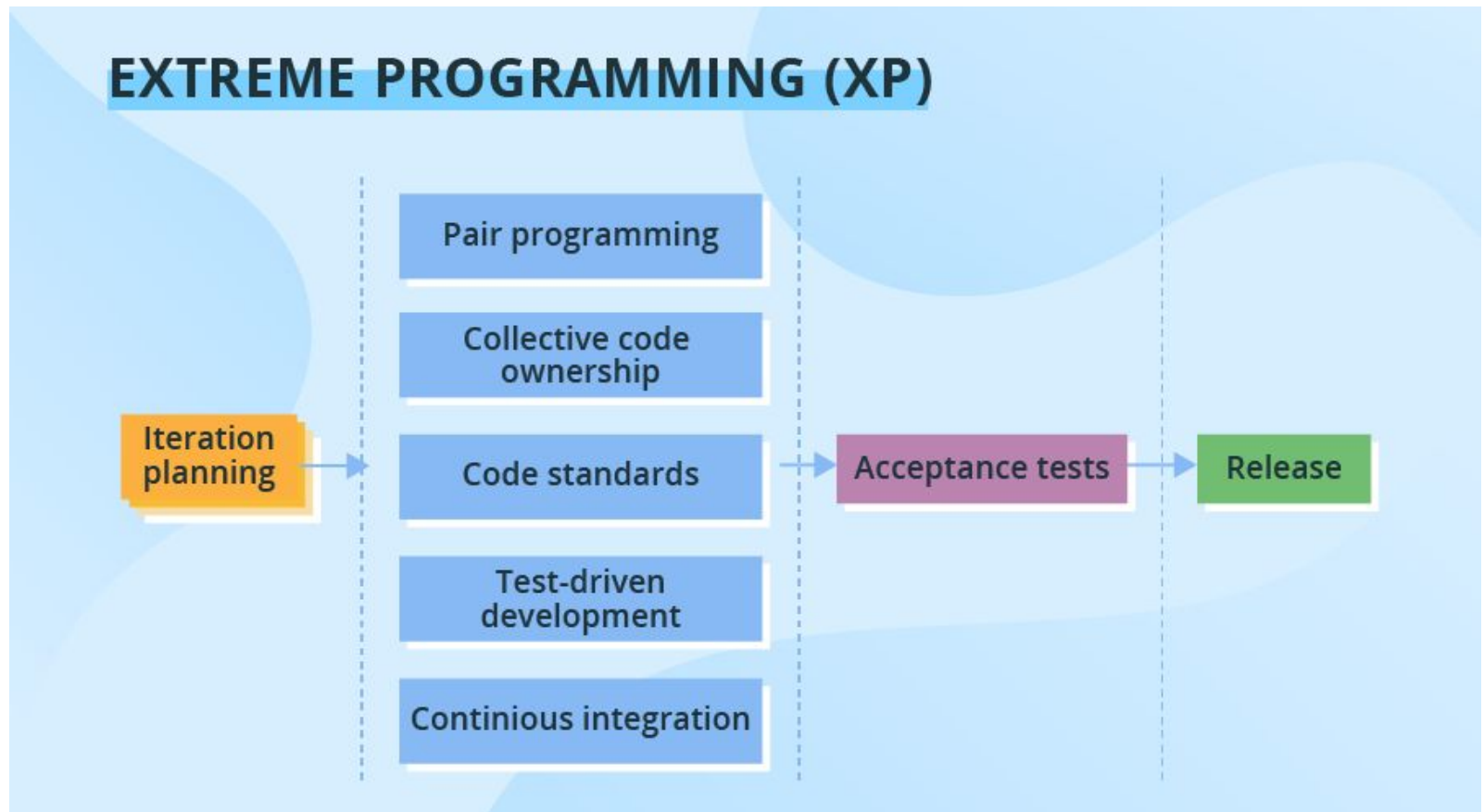
## THE RATIONAL UNIFIED PROCESS (RUP)



# Modelo ágil - Scrum

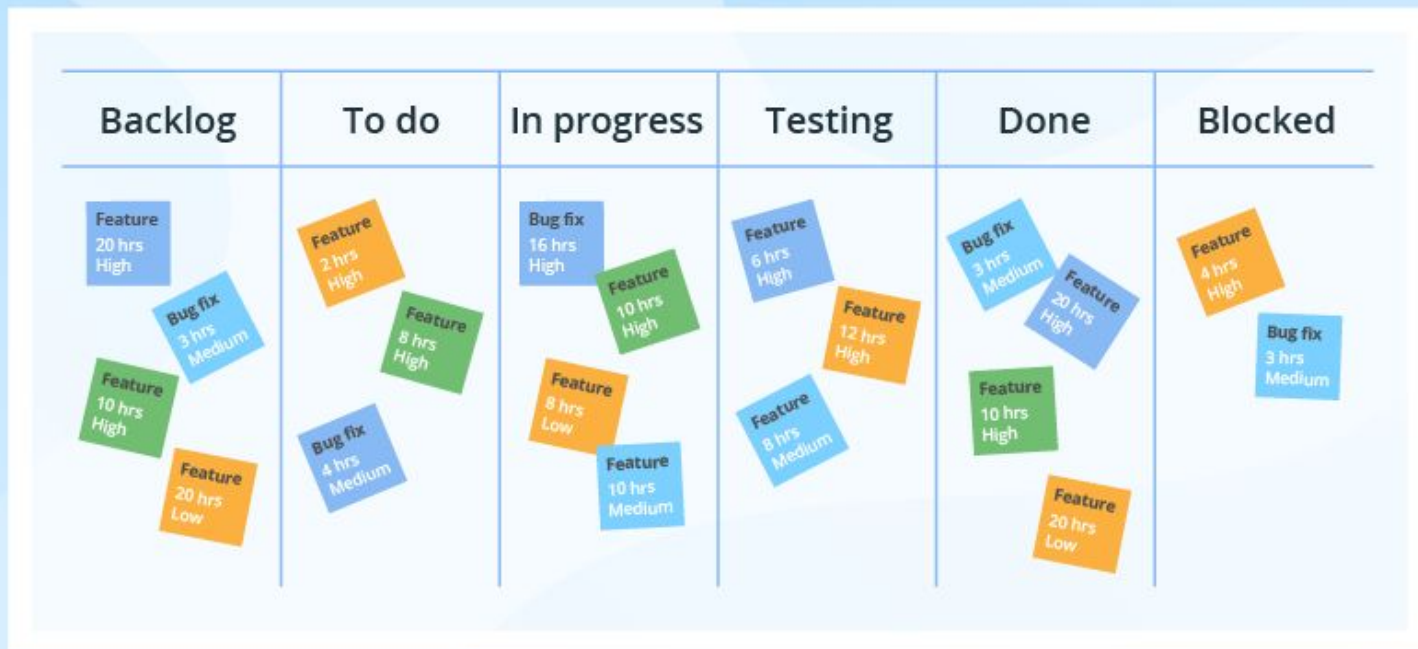


# Modelo ágil - XP



# Modelo ágil - Kanban

## KANBAN



# Proceso de prueba



# Proceso de prueba

No existe uno universal...

.... pero hay actividades/tareas de prueba comunes.

- Planificación de la prueba
- Monitorización (seguimiento) y control de la prueba
- Análisis de la prueba
- Diseño de la prueba
- Implementación de la prueba
- Ejecución de la prueba
- Compleción de la prueba

# Proceso de prueba - Niveles

Cada nivel = instancia del proceso de prueba

- se relaciona a etapa específica del ciclo de vida del software
- se enfoca en diferentes aspectos del software en desarrollo

## **Niveles:**


- Prueba de componente (unitaria)
- Prueba de integración
- Prueba de sistema
- Prueba de aceptación

# Planificación

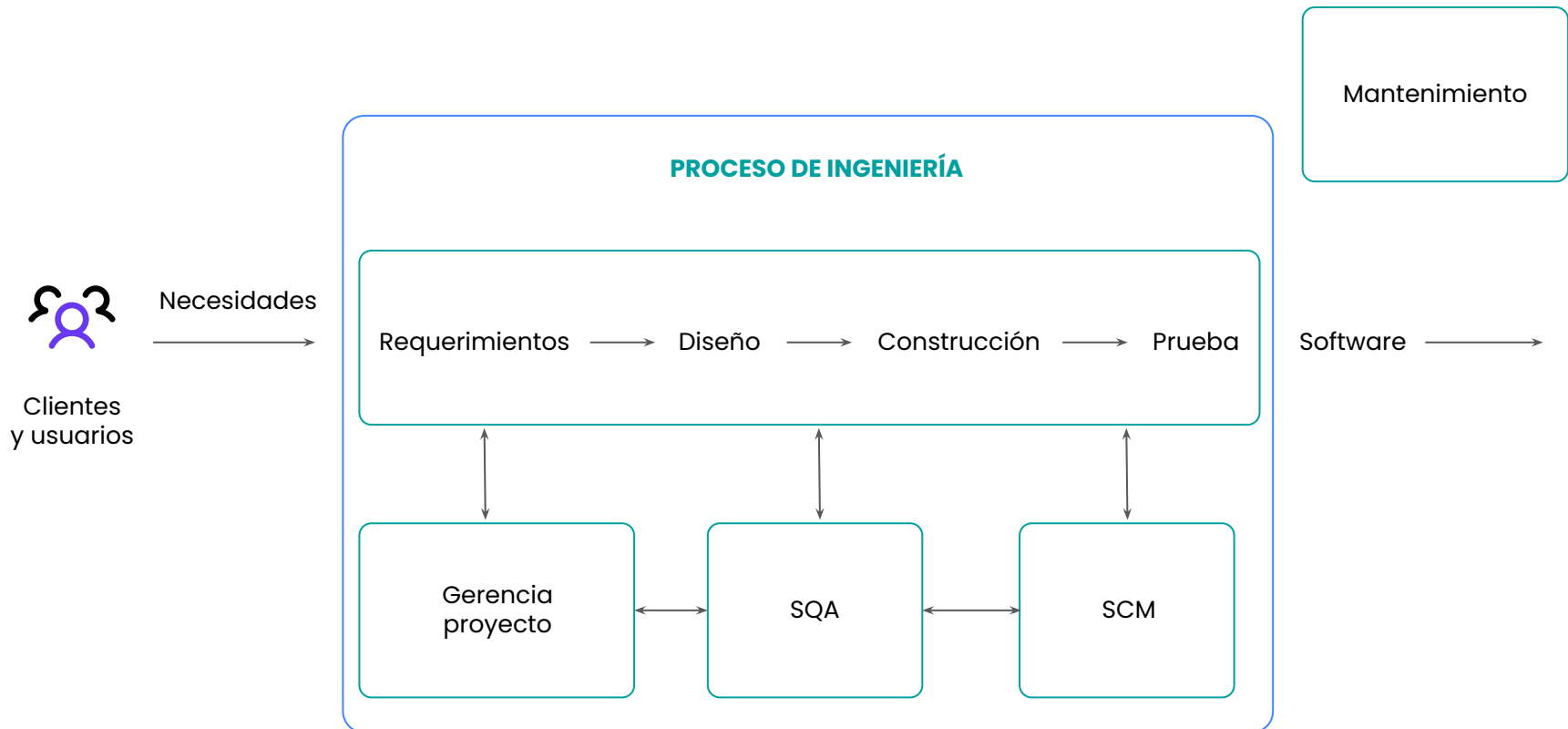
## Plan de pruebas

- Alcance, objetivos y riesgos de la prueba
- Enfoque general de la prueba
- Integrar/coordinar actividades de prueba en el SDLC
- Quién y cómo se realizarán las pruebas
- Calendario (cuándo). Fechas o momento en iteración
- Métricas para monitorizar y controlar
- Presupuesto
- Nivel de detalle y estructura de de doc. de prueba

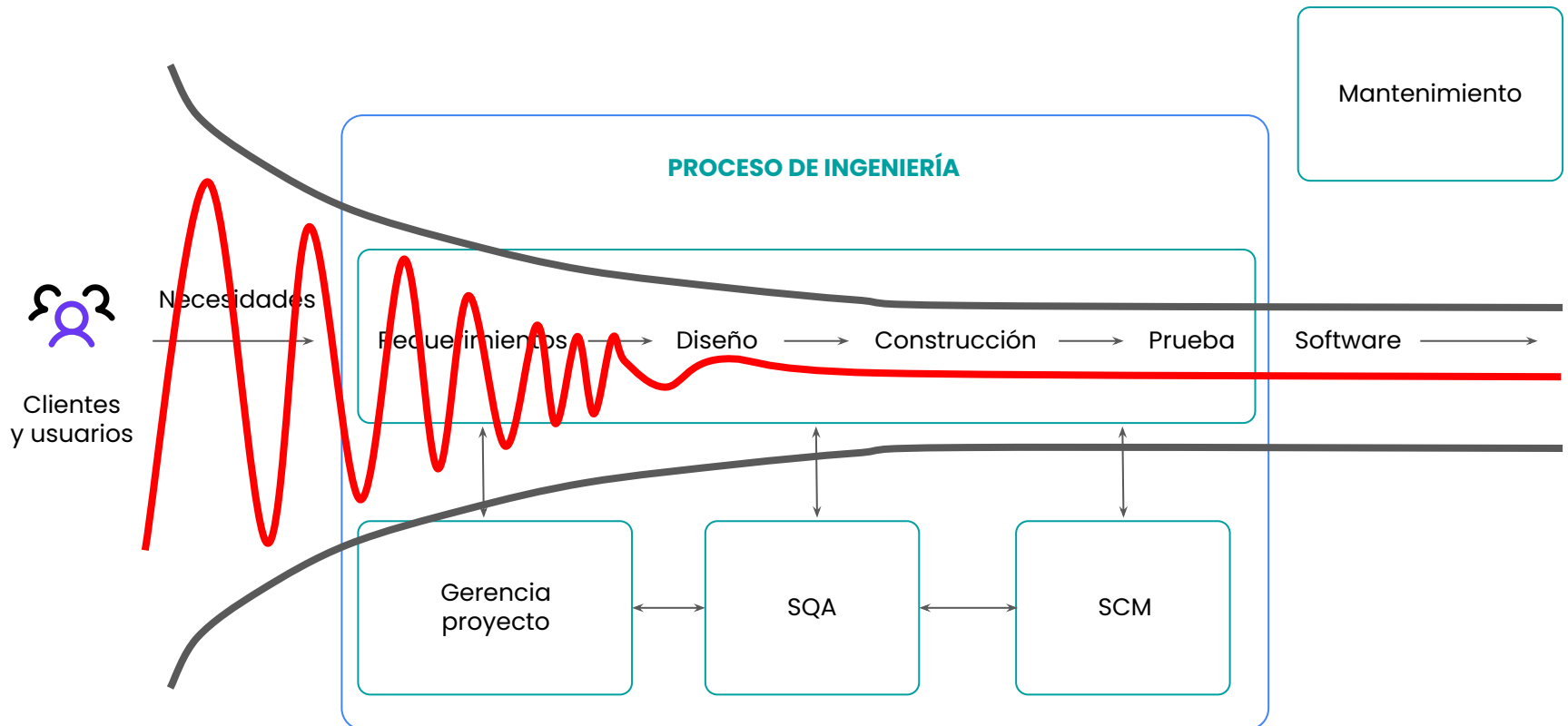
# Planificación - Ej. prueba unitaria

1. **Objetivos de las pruebas:** Las pruebas unitarias tienen como objetivo detectar errores en el código fuente y asegurarse de que cada unidad de código se comporta de acuerdo a lo esperado.
2. **Alcance de las pruebas:** Las pruebas unitarias cubrirán todas las funciones, procedimientos y clases del código fuente.
3. **Estrategia de pruebas:** Las pruebas unitarias se realizarán utilizando una herramienta de pruebas automatizadas y se ejecutarán en el ambiente de desarrollo del software.
4. **Criterios de aceptación:** Se considerará que las pruebas unitarias son exitosas si todas las pruebas pasan satisfactoriamente y si se logra una cobertura de código del 100%. 
5. **Tareas y responsabilidades:**
  - El equipo de desarrollo es responsable de crear las pruebas unitarias.
  - El equipo de pruebas es responsable de revisar las pruebas unitarias y asegurarse de que cubren todas las funcionalidades del código.
  - El equipo de pruebas es responsable de la ejecución de las pruebas unitarias y la documentación de los resultados.
6. **Calendario de pruebas:** Las pruebas unitarias se realizarán durante la fase de desarrollo del software y se ejecutarán después de que se haya completado la implementación de cada unidad de código.
7. **Riesgos y contingencias:** Los riesgos incluyen la posibilidad de que no se logre una cobertura de código completa o de que se identifiquen defectos críticos en el código. La contingencia consistirá en realizar una revisión adicional de las pruebas y de los resultados para asegurarse de que se cumplan los objetivos.
8. **Recursos necesarios:** Serán necesarios un equipo de pruebas, una herramienta de pruebas automatizadas y acceso al ambiente de desarrollo del software.
9. **Resultados esperados:** Se espera que todas las pruebas unitarias pasen satisfactoriamente y que se documenten los resultados de las pruebas y los defectos encontrados.
10. **Entregables:** Los entregables incluirán la documentación de los resultados de las pruebas, los defectos encontrados y las correcciones realizadas. También se archivarán los casos de prueba y los resultados de las pruebas para futuras referencias.

# Monitorización (seguimiento) y control de la prueba



# Monitorización (seguimiento) y control de la prueba



---

# Monitorización (seguimiento) y control de la prueba

## Actividades:

- Seguimiento del progreso de las pruebas
- Medición y reporte de resultados
- Gestión de riesgos
- Actualización del plan de prueba
- Comunicación y colaboración

# Análisis

Busca analizar la base de la prueba y definir los objetivos de la prueba (**qué probar**)

Pasos:

- Identificar la base de la prueba
- Analizar la base de la prueba
- Definir los objetivos de la prueba



# Análisis - Ej. prueba unitaria

- La base de la prueba es la especificación de requisitos del software.
- El analista de pruebas revisa la especificación de requisitos para comprender la funcionalidad del software.
- El analista de pruebas identifica las siguientes condiciones de prueba:
  - El software debe ser capaz de sumar dos números.
  - El software debe ser capaz de restar dos números.
  - El software debe ser capaz de multiplicar dos números.
  - El software debe ser capaz de dividir dos números.
- El analista de pruebas prioriza condiciones de prueba en función del riesgo. Más riesgo → más probabilidades de causar un defecto. En este caso, la condición de prueba más arriesgada es la que comprueba la capacidad del software para dividir dos números.
- El analista de pruebas diseña casos de prueba de alto nivel para cada condición de prueba (suma, resta y multiplicación, relativamente sencillos; división es más complejo por división entre cero).

# Diseño

Transformar condiciones de prueba en:

- casos de prueba
- conjuntos de casos de prueba
- otros productos de prueba

Responde a la pregunta de “**cómo probar**”

# Diseño

## Actividades:

- Diseñar/priorizar casos de prueba
  - uso de técnicas de pruebas
- Identificar datos de prueba necesarios
  - *“test oracle”*
- Diseñar entorno de prueba

# Implementación

Secuenciar casos de prueba en procedimientos de prueba

Actividades:

- Desarrollar guiones de prueba (automatizados?)
- Crear juegos de prueba
- Armar un calendario de ejecución
- Construir entorno de prueba
- Preparar datos de prueba y cargarlos al entorno

# Ejecución

Juegos de prueba se ejecutan según calendario

Actividades:

- Versionar elementos de prueba
- Ejecutar pruebas
- Comparar resultados reales vs. esperados
- Establecer causas probables de anomalías
- Registrar resultados e informar defectos
- Repetir pruebas

# Compleción

Recopilar datos y consolidar la experiencia. En ágil se da al finalizar una iteración.

## Actividades:

- Comprobar que informes de defectos están cerrados
- Finalizar, archivar y almacenar todo
- Traspaso de productos de prueba a mantenimiento
- Analizar lecciones aprendidas
- Mejorar proceso de prueba

# El proceso en contexto

# El proceso en contexto

Factores que influyen en proceso de prueba:

- Modelo de ciclo de vida
- Niveles de prueba (componente, integración, sistema, aceptación)
- Tipos de prueba (funcional/no funcional, caja blanca/negra)
- Dominio de negocio
- Restricciones operativas (recursos, plazos, complejidad, normas)
- Políticas de la organización
- Estándares internos y externos



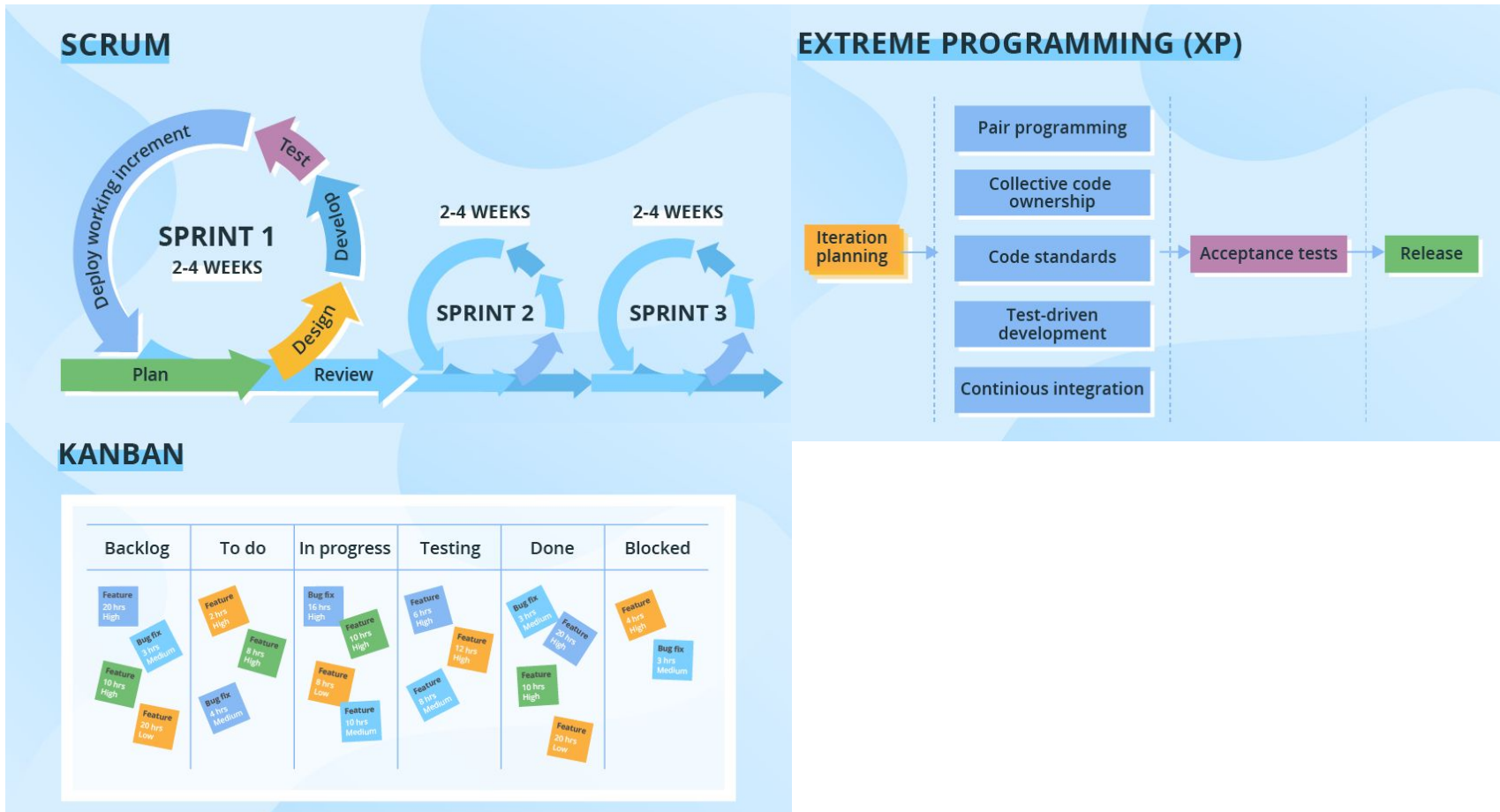
# El proceso en contexto

Factores que influyen en proceso de prueba:

- **Modelo de ciclo de vida**
- Niveles de prueba (componente, integración, sistema, aceptación)
- Tipos de prueba (funcional/no funcional, caja blanca/negra)
- **Dominio de negocio**
- **Restricciones operativas** (recursos, plazos, complejidad, normas)
- Políticas de la organización
- Estándares internos y externos

# El proceso en contexto

## - Modelo de ciclo de vida



# El proceso en contexto

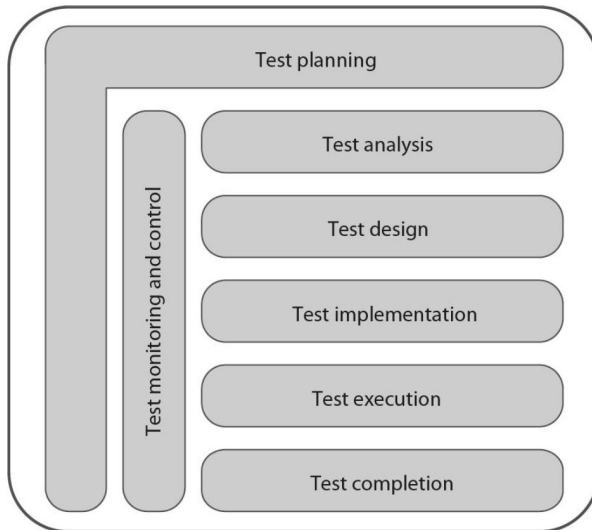


Fig. 2-3 The testing process

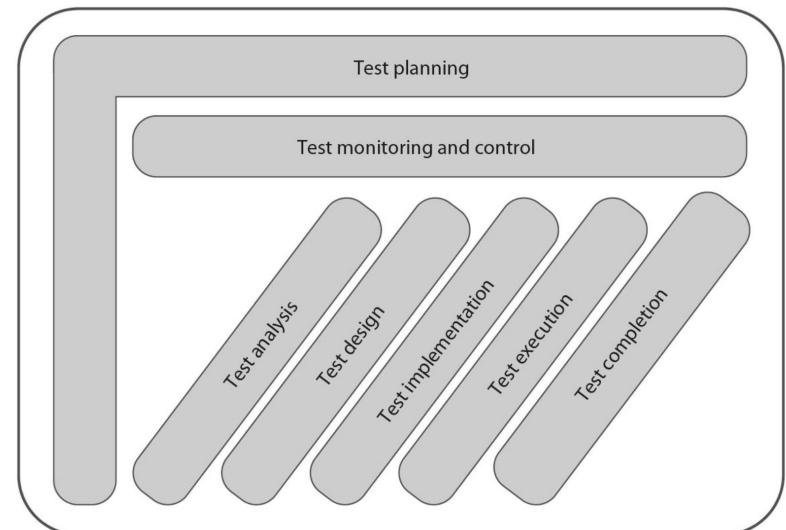


Fig. 2-4 The test process showing time overlap

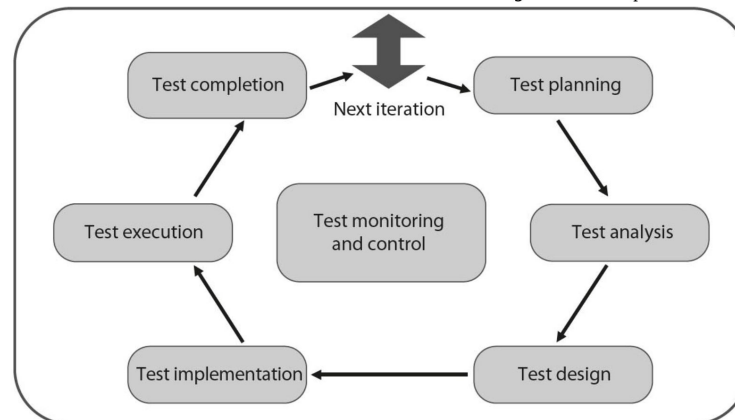


Fig. 2-5 An iterative test process

# El proceso en contexto

## Dominio de negocio

- Relevancia: pruebas sobre aspectos relevantes/críticos
- Cobertura: evitar omisión de pruebas importantes
- Comprensión: identificar comportamiento esperado
- Comunicación: lenguaje común con devs y cliente

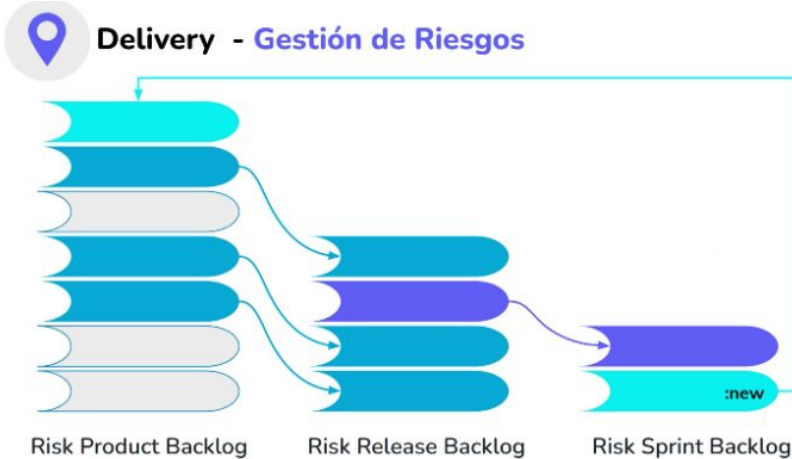
# El proceso en contexto

## Restricciones operativas: recursos limitados (Startups?)

- Enfoque ágil
- Automatización de pruebas
- Feedback rápido
- Priorización selectiva
  - Priorizar pruebas en función de
    - riesgos
    - áreas claves del negocio

# El proceso en contexto

## Paréntesis sobre riesgos

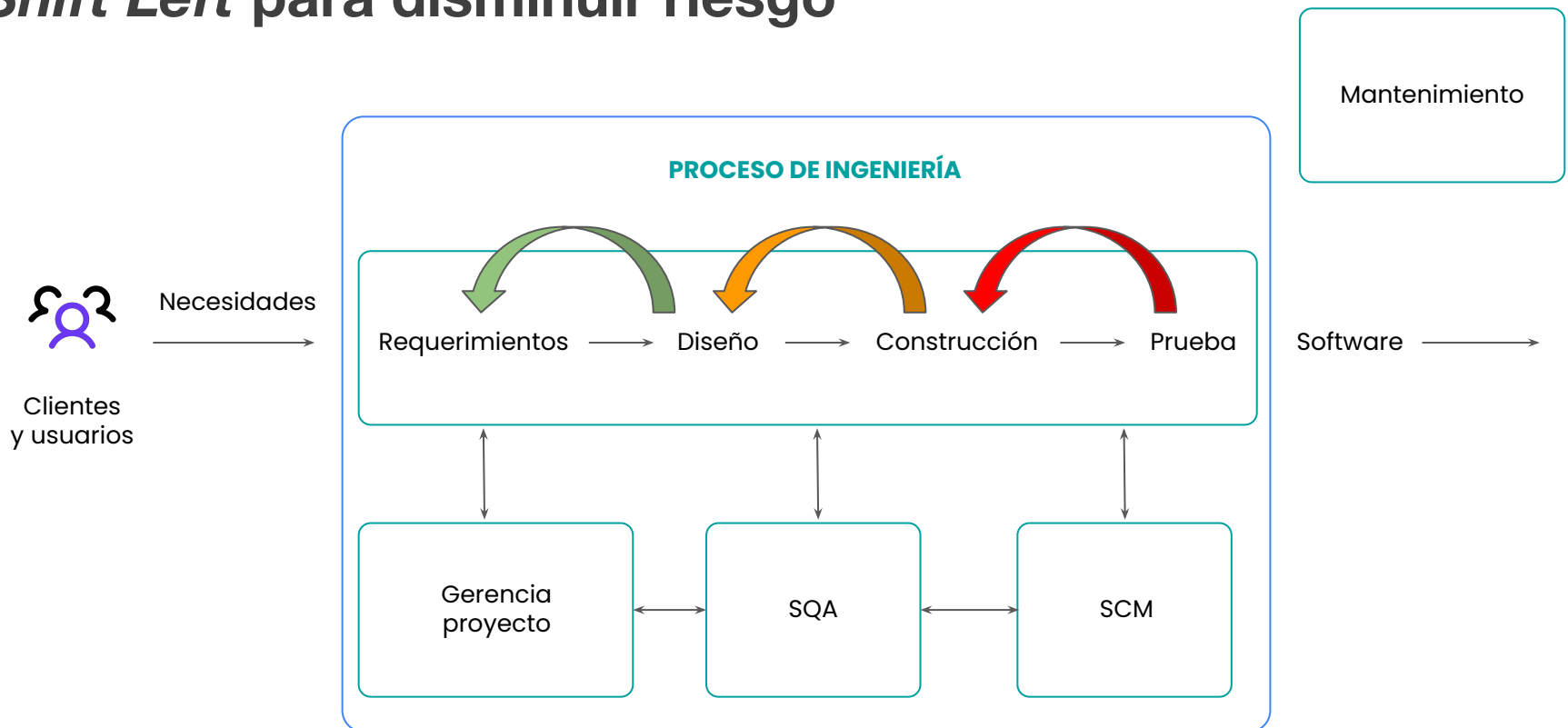


### #R002: Disponibilidad de agenda del cliente

<b>Descripción:</b>	<b>Acción preventiva:</b>
Dado que el cliente no nos puede atender . Cuando hacemos el relevamiento. Entonces eso nos puede retrasar el proyecto.	Coordinación de reuniones con bastante anticipación.
<b>Disparador:</b>	<b>Acción correctiva:</b>
El cliente demora más de 2 semanas en contestar una consulta.	En última instancia de que el cliente deje de responder, el equipo se apropiaría del proyecto y lo continuaría.
<b>Estrategia:</b> Aceptar.	

# El proceso en contexto

## *Shift Left* para disminuir riesgo



# El proceso en contexto

## *Shift Left* para disminuir riesgo

Involucrar *testers* en etapas tempranas para:

- Aprender: *testing* también es evaluar un producto **aprendiendo** de él mediante la exploración y la experimentación.
- Desafiar otros roles: agregar escepticismo, anticipar problemas (*Doomsayer* de XP?) y encontrar puntos donde nos estemos “engañando” utilizando pensamiento crítico.
- Perseguir la *testeabilidad*
- *Pair testing*: desarrollador explica el código al tester. Luego este puede hacer preguntas y finalmente probar la funcionalidad.