FACULTAD DE INGENIERIA

# Fundamentos de Ingeniería de Software

## Procesos de desarrollo de software

María de las Nieves Freira

Universidad de la República

# Temario

- Introducción
- Modelos de proceso de software
- Herramientas y técnicas
- Modelo de proceso RUP
- Procesos ágiles

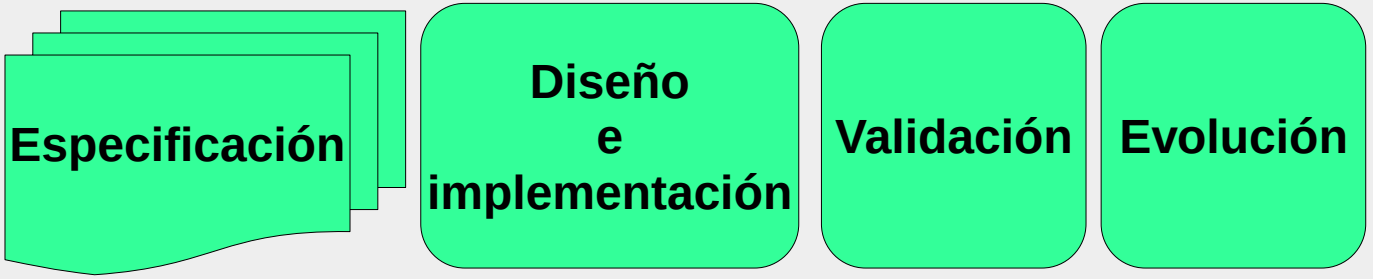
# Proceso

Serie de pasos con:

- Actividades
- Restricciones
- Recursos

que producen una salida de cierto tipo.

Procesos de software involucran



# Modelo de proceso

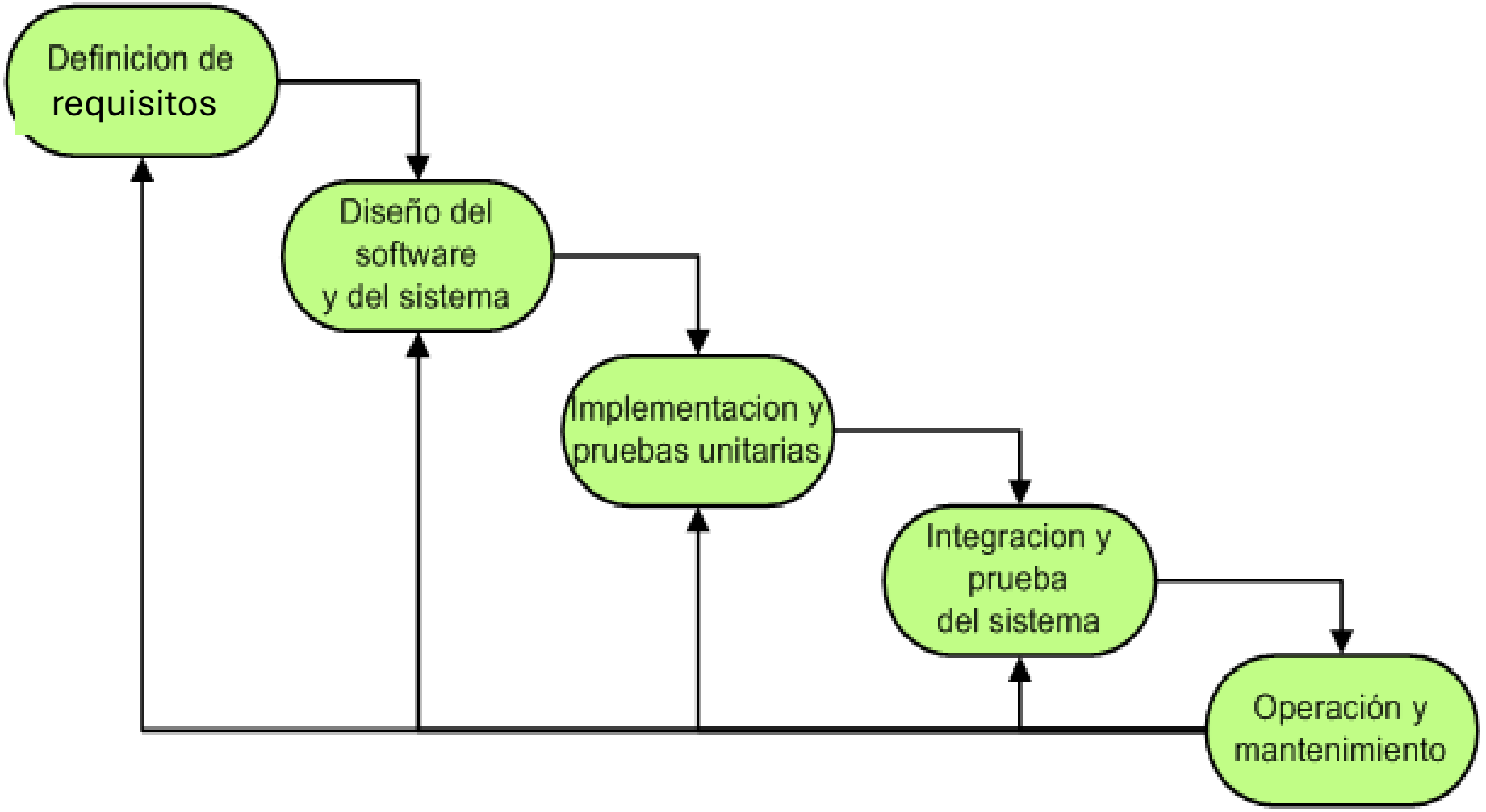
Modelo de proceso de software:

- representación abstracta de un proceso
- descripción a partir de una perspectiva en particular

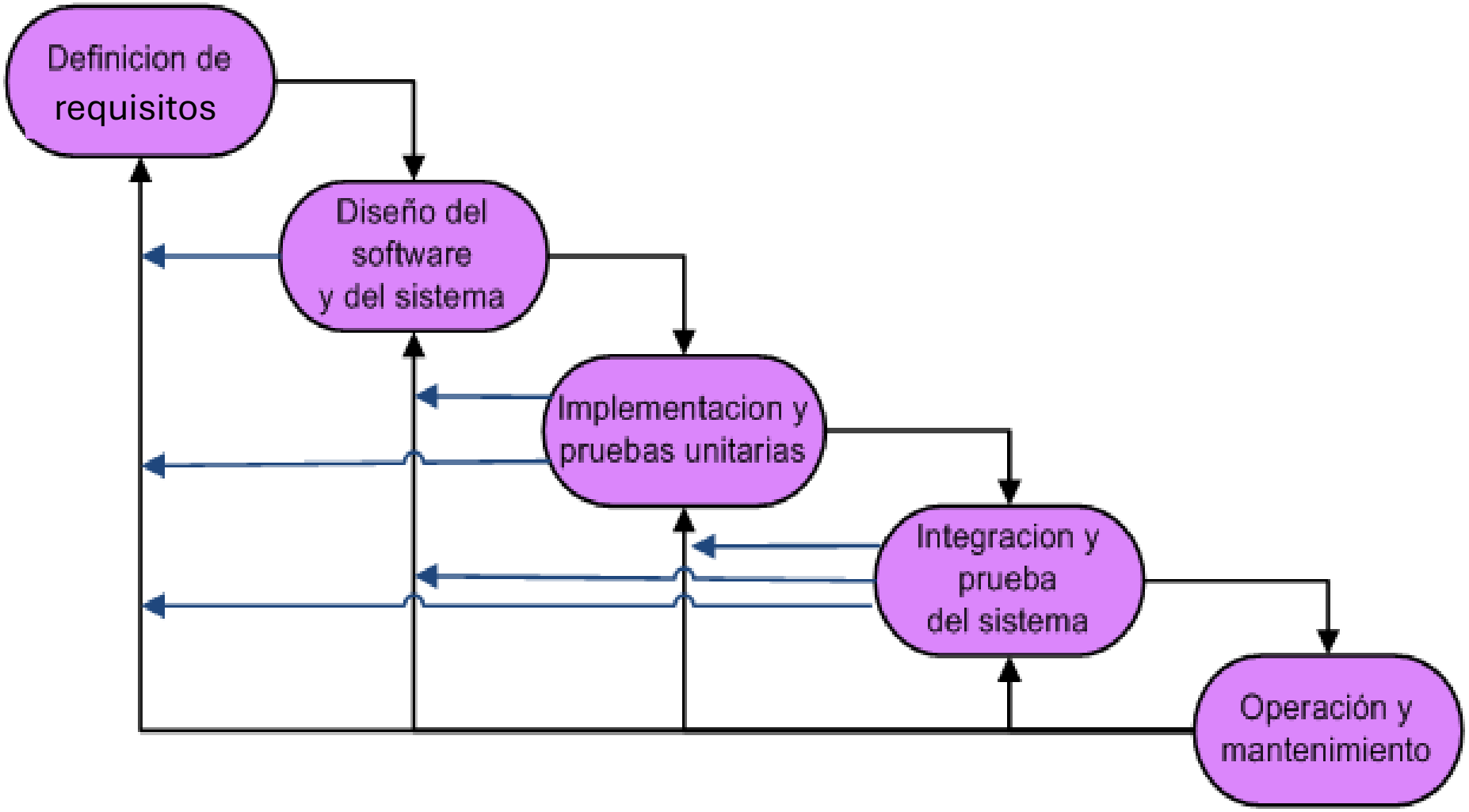
Pueden usarse como:

- Prescripciones de la forma en que el desarrollo de software **debería llevarse a cabo**
- Descripciones de la forma en que el desarrollo **se lleva a cabo realmente**

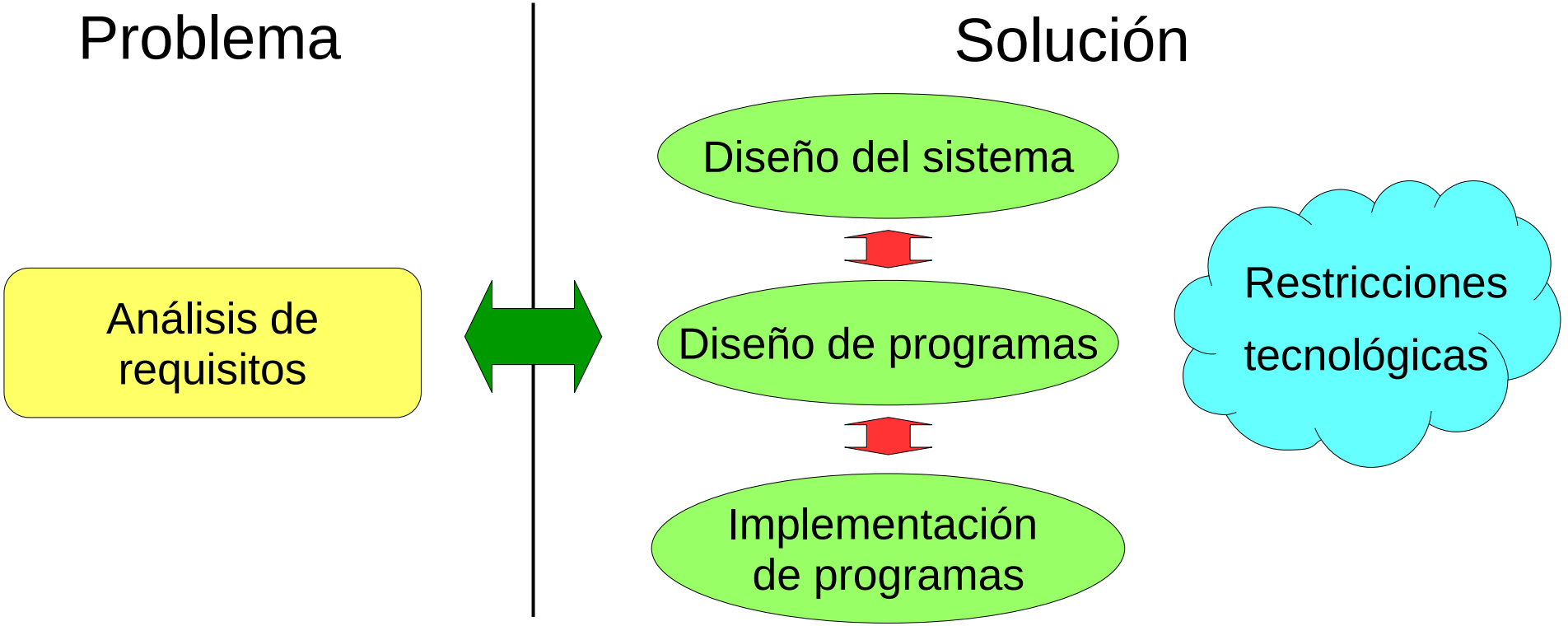
# Modelo en cascada



# Lo que realmente ocurre...



# Espacios problema y de la solución

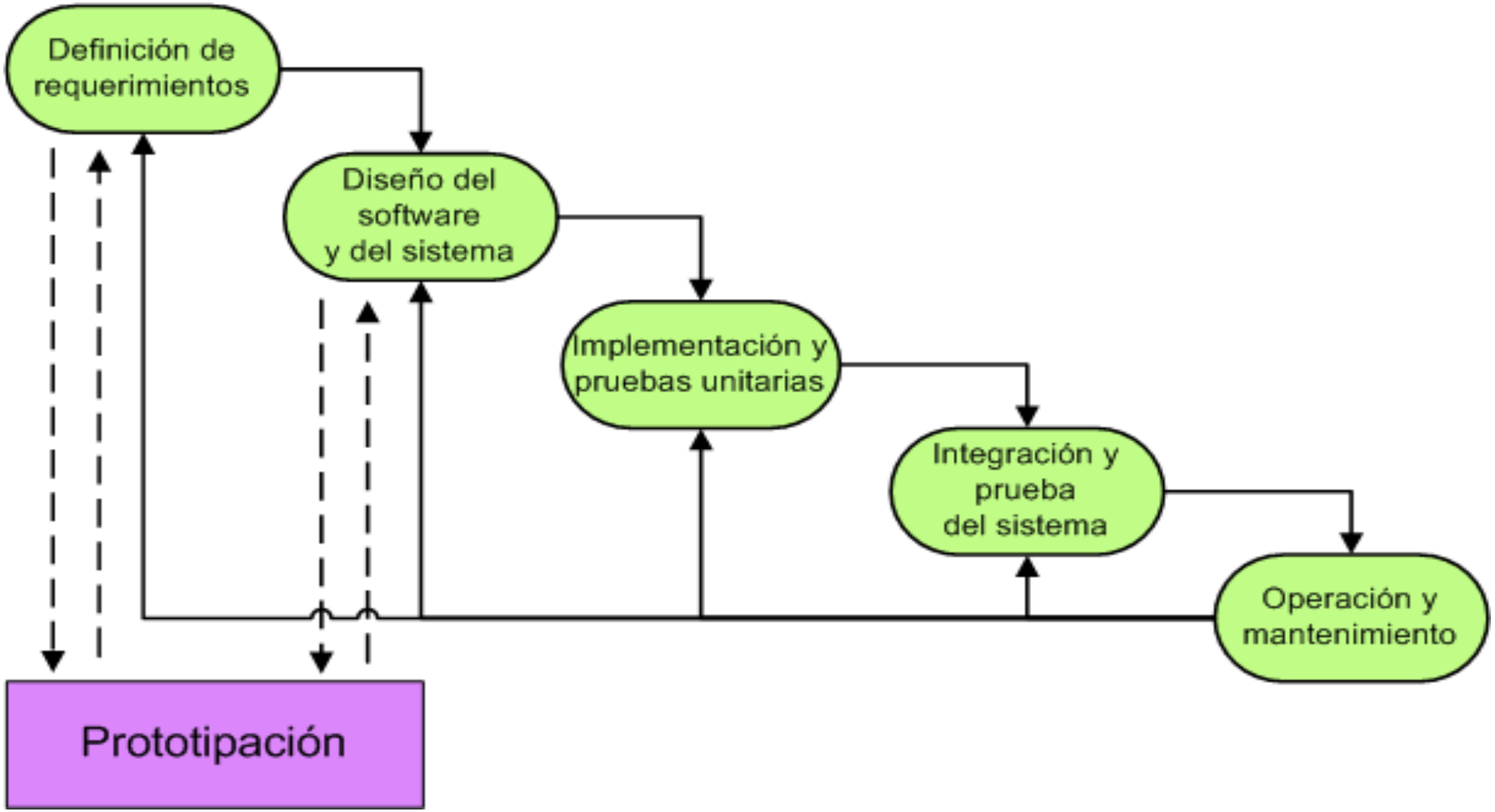



- Intentar solucionar un problema a menudo aclara el problema
- Evaluar los atributos de calidad externos del software (solución) requiere que exista «comportamiento»

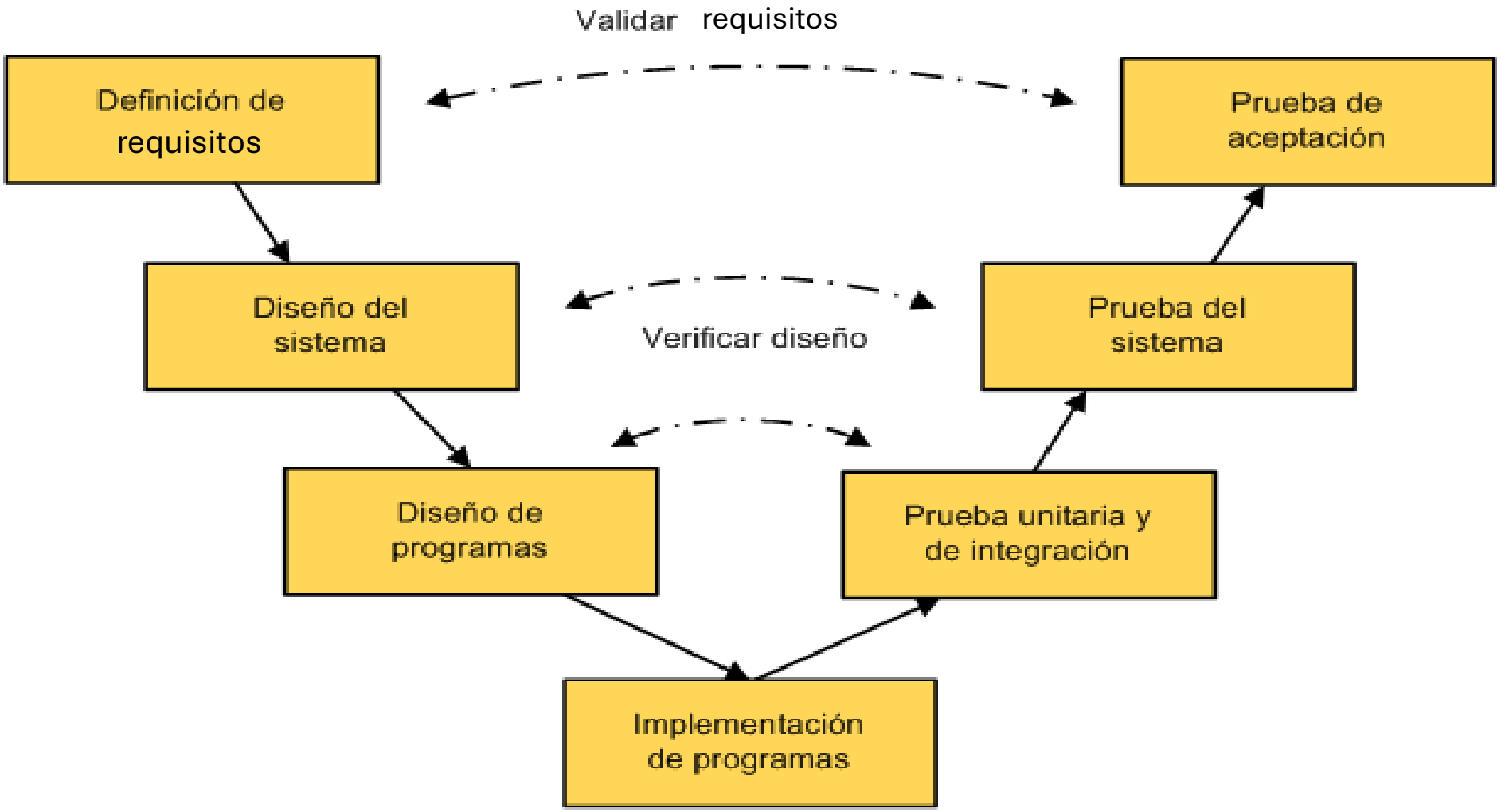
# Problemas del proceso en cascada

- Riesgo de que, al ver el producto funcionando, el cliente diga: «No es lo que preciso».
- Requisitos
  - ¿Cuándo están completos los requisitos?
  - ¿Serán consistentes?
- No permite manejar bien cambios en los requisitos.
  - ¿Por qué pueden cambiar?
  - El costo del cambio es mayor a medida que avanza el proyecto.
- Entrega tardía de valor

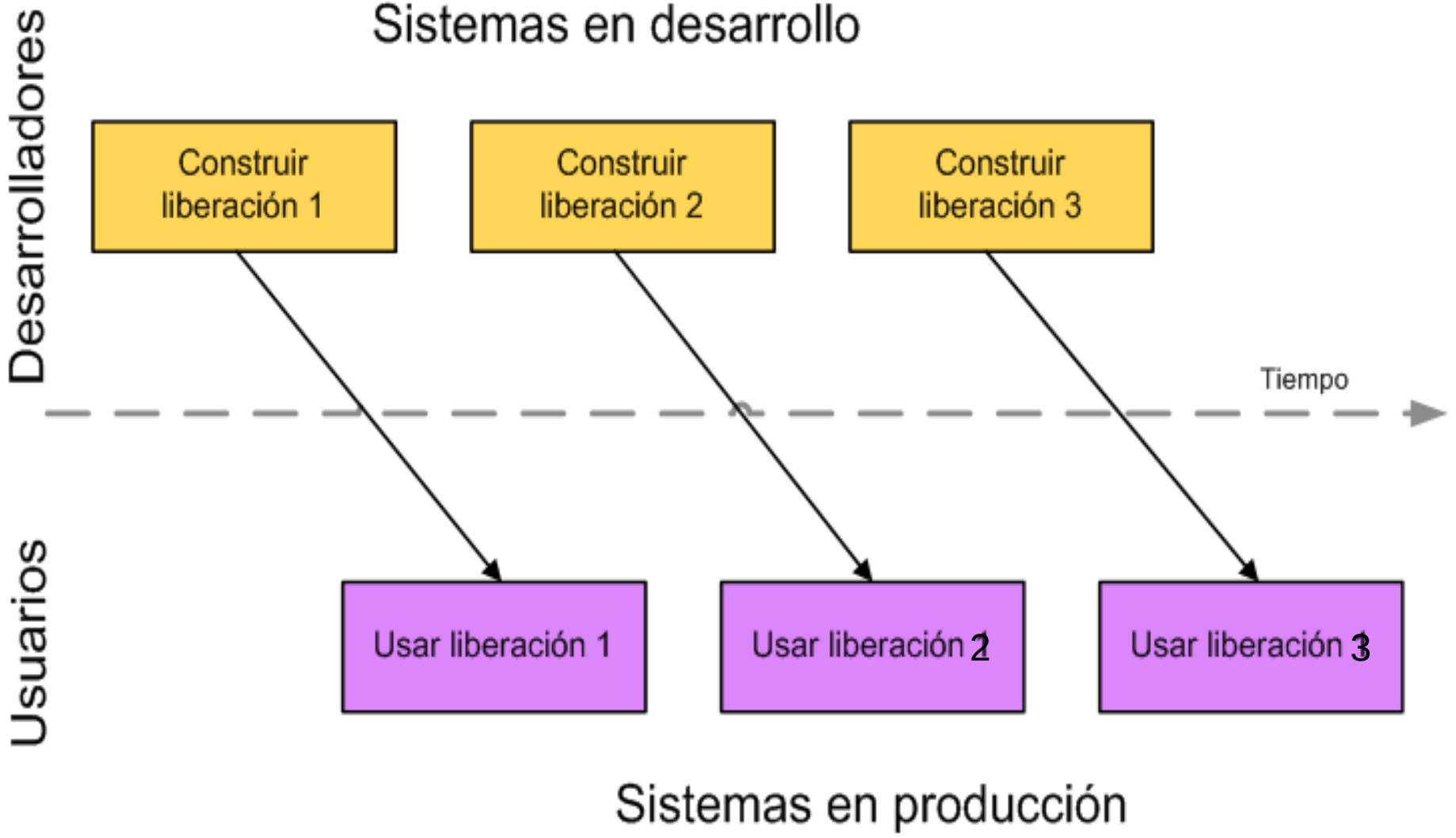
# Cascada con prototipación



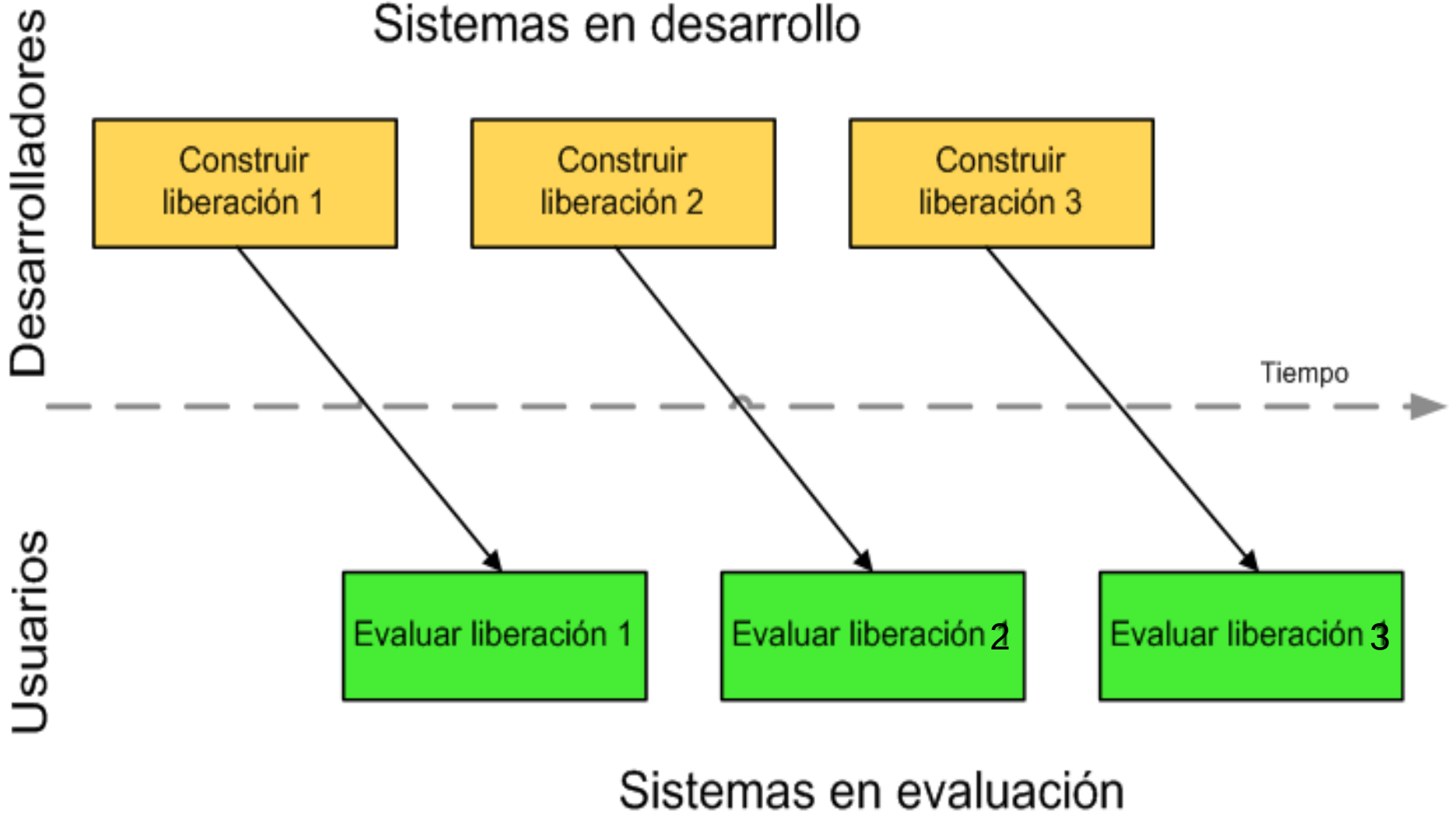
# Modelo V


Validar requisitos
Definición de requisitos
Prueba de aceptación
Diseño del sistema
Verificar diseño
Prueba del sistema
Diseño de programas
Prueba unitaria y de integración
Implementación de programas

# En fases con liberaciones parciales

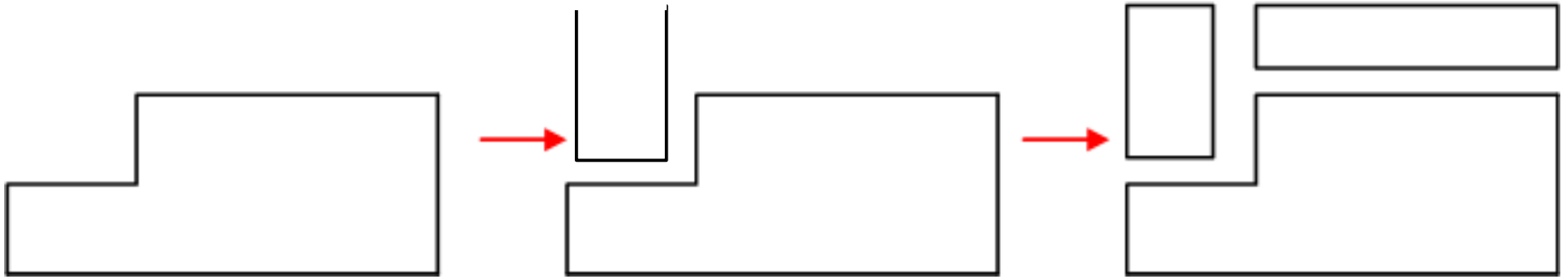


# En fases con evaluaciones parciales

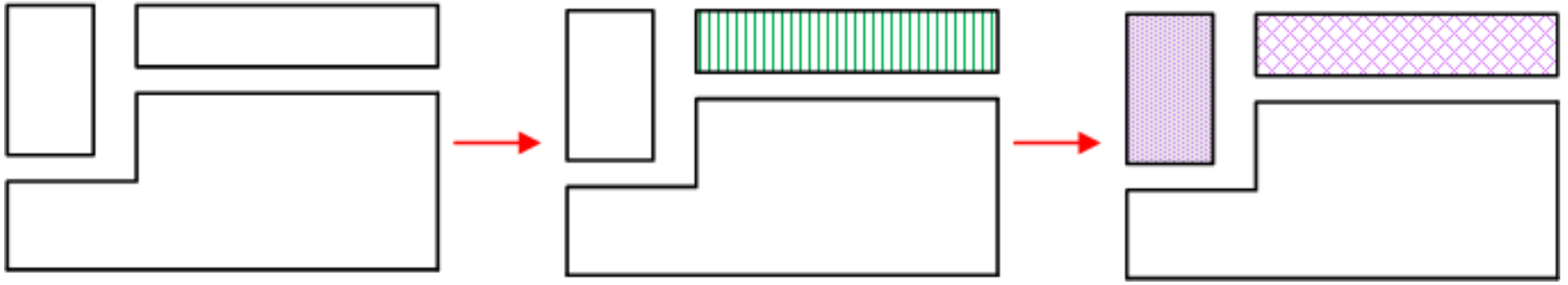


# Incrementos e iteraciones

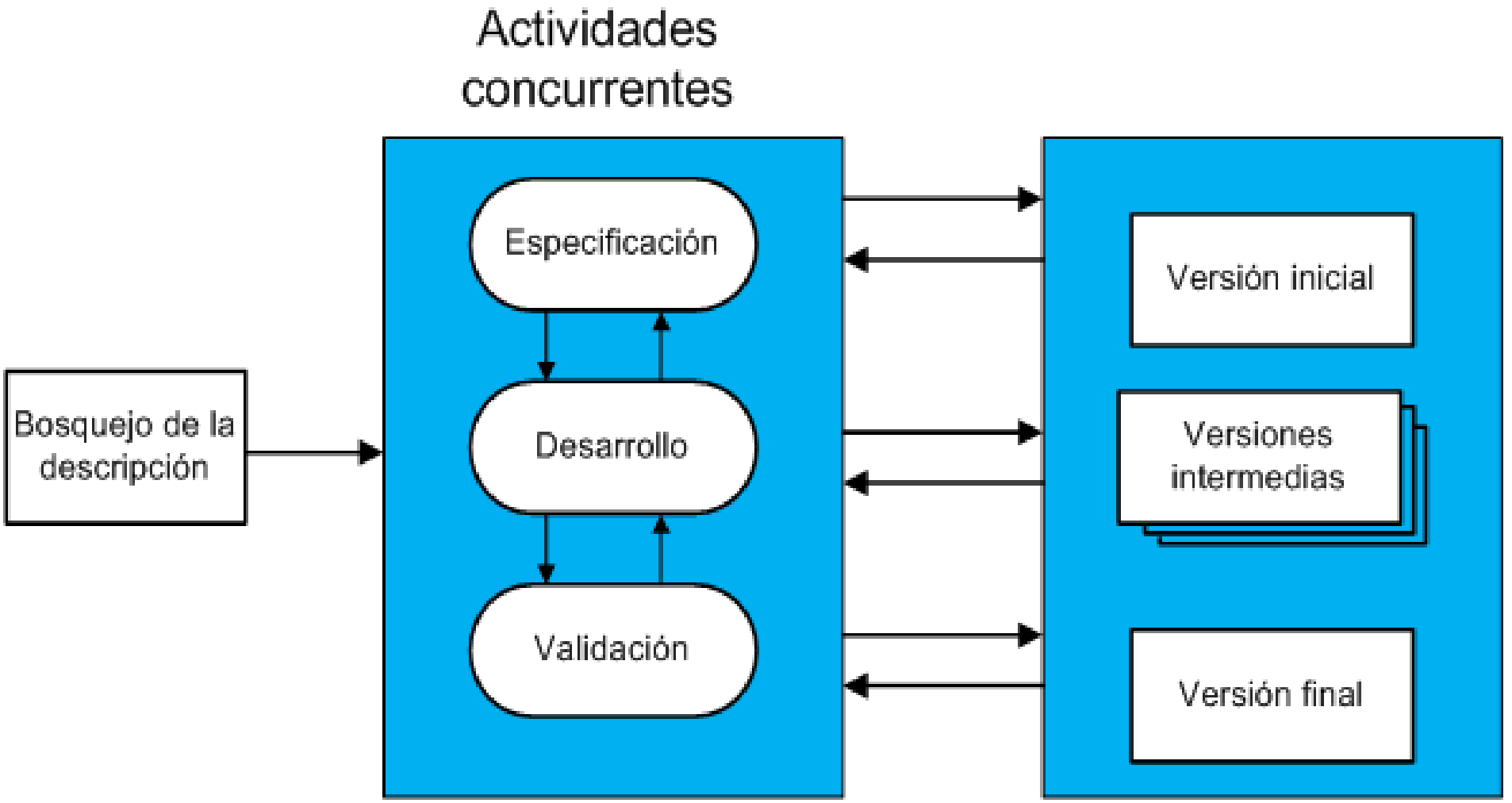
Desarrollo incremental



Desarrollo iterativo



# Desarrollo incremental



# Herramientas y técnicas para el modelado de procesos

- Elegir un lenguaje o notación
- Tener claro los objetivos del modelo
  - Detalle (granularidad)
  - Describir-prescribir
  - Predecir (requiere agregar relaciones cuantitativas entre elementos)
  - Ejecutar (asistir en el uso)

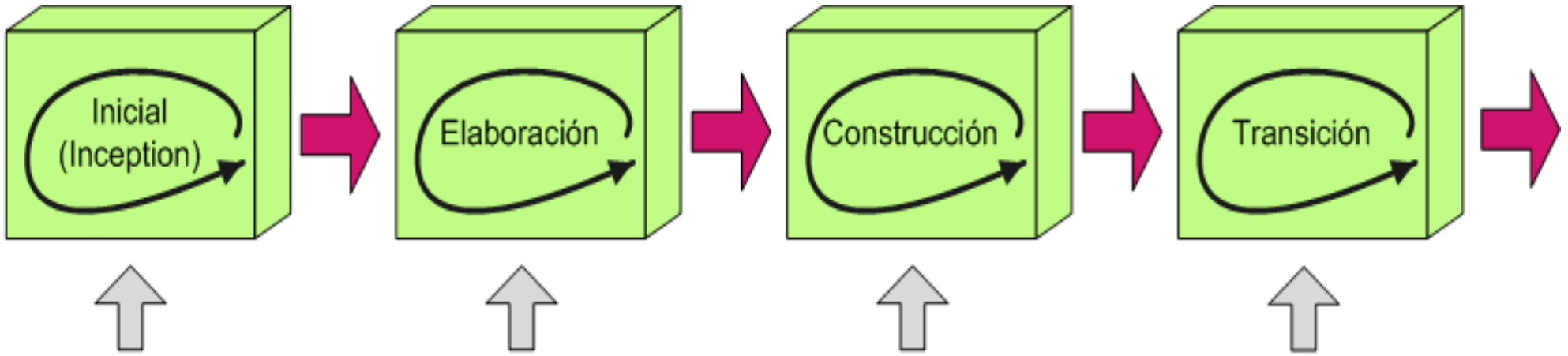
# Notaciones y herramientas

- UML *(Unifed Modeling Language)*
  - Editores UML
- BPMN *(Business Process Model and Notation)* – Especificación de Procesos (no necesariamente de software)
  - Editores BPMN
- SPEM 2 (Software & Systems Process Engineering Metamodel™ Specification)
  - Eclipse Process Framework Composer

# RUP – Rational Unified Process

- Desarrollo en fases
- Cada fase con objetivos definidos
- Iterativo
  - Cada fase compuesta de iteraciones
  - Todas las fases se pueden repetir
- Se describe desde tres perspectivas:
  - Dinámica – en el tiempo
  - Estática – Flujos de trabajo (disciplinas)
  - Buenas prácticas
- Centrado en la arquitectura

# Fases del RUP




Establecer el modelo de negocio para el sistema.

Desarrollar un entendimiento del dominio del problema y la arquitectura del problema.

Diseño del sistema, programación y testing.

Deploy del sistema en su entorno operativo.

# Flujos de trabajo (disciplinas) (I)

| Flujo | Descripción |
|---|---|
| Modelos de negocio | Los procesos de negocio se modelan mediante casos de uso del negocio. |
| Requisitos | Se identifican los actores que interactúan con el sistema y se han desarrollado los casos de uso para modelar los requisitos del sistema. |
| Análisis y diseño | Se crea un modelo de diseño y se documenta usando modelos de arquitectura, modelos de componentes, modelos de objetos y modelos de secuencia. |
| Implementación | Se implementan los componentes del sistema y estructurados dentro de los subsistemas. La generación automática de código a partir de los modelos de diseño ayuda a acelerar estos procesos. |

# Flujos de trabajo (disciplinas) (2)

| Flujo | Descripción |
|---|---|
| Testing | El testing es un proceso iterativo que se realiza en conjunto con la aplicación. |
| Deployment | Se crea una versión del producto, que es distribuido a los usuarios e instalado en su lugar de trabajo. |
| Configuración y gestión de los cambios | Este flujo de trabajo gestiona cambios en el sistema. |
| Gestión de proyectos | Este flujo de trabajo gestiona el desarrollo del sistema. |
| Entorno | Este flujo de trabajo se ocupa de la disposición de herramientas de software adecuadas al equipo de desarrollo de software. |

# RUP – Buenas prácticas

- Desarrollar software iterativamente
  - Los incrementos del proyecto se basan en las prioridades del cliente por lo cual se liberan primero las de mayor prioridad.
- Gestión de requisitos
  - Documentar de forma explícita los requerimientos del cliente y llevar un seguimiento de los cambios en los requisitos.
- Utilizar las arquitecturas basadas en componentes
  - Organizar la arquitectura del sistema como un conjunto de componentes reutilizables.

# RUP – Buenas prácticas (2)

- Visualización del modelo de software
  - Usar modelos gráficos en UML para presentar vistas estáticas y dinámicas del software.
- Verificar la calidad del software
  - Asegurarse que se cumplen las normas de calidad de la organización.
- Controlar los cambios en el software
  - Administrar los cambios en el software utilizando un sistema de gestión de cambios y las herramientas de gestión de la configuración.

# Hagamos al revés...

- Fijemos el tiempo
  - Dados los recursos
  - Veamos qué alcance logramos
- Con fases de corta duración
  - La presión del cronograma está siempre presente
  - El alcance normalmente tiene relevancia dispar (principio de Pareto) y el equipo puede poner foco en lo más relevante
  - Olvidarse de los cambios
    - Los cambios se pueden incluir en una fase siguiente

# Procesos guiados por planes y ágiles

- Procesos guiados por planes (pesados):
  - Cascada, RUP, MUM
  - El plan se construye fundamentalmente al comienzo
  - La comunicación se basa en documentos
- Procesos Ágiles:
  - XP, SCRUM, Kanban, TDD, FDD
  - El plan se completa a medida que avanza el proyecto
  - La comunicación está basada en conversaciones cara a cara

# Manifiesto por el desarrollo ágil (2001)

Hemos aprendido a valorar:

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

aunque valoramos los elementos de la derecha, valoramos más los de la izquierda.

# Principios del Manifiesto Ágil

- Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor.
- Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos Ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.
- Entregamos software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible.
- Los responsables de negocio y los desarrolladores trabajamos juntos de forma cotidiana durante todo el proyecto.
- 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.
- El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.
- El software funcionando es la medida principal de progreso.
- Los procesos Ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante de forma indefinida.
- La atención continua a la excelencia técnica y al buen diseño mejora la Agilidad.
- La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial.
- Las mejores arquitecturas, requisitos y diseños emergen de equipos autoorganizados.
- A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo para a continuación ajustar y perfeccionar su comportamiento en consecuencia.

# Time-boxing

En gestión de proyectos tenemos la triple restricción:


- Cambiar una afecta el resto de forma poco predecible.
- Los proyectos se gestionan a menudo por alcance fijo
  - Si no alcanza el plazo, se extiende o se agregan recursos
- Bueno trabajar en fases, pero …
  - Difícil estimar cuánto debe durar cada una para cierto alcance
  - Si se atrasa ¿cuánto permitiremos que se atrase?

# SCRUM

- Roles
  - Principales
    - *Product owner* (representante de los involucrados)
    - Equipo de desarrollo
    - *Scrum master*
  - Auxiliares
    - *Stakeholders* (clientes-vendedores)
    - Gerentes
- Sprint (1 semana a 1 mes – time-boxing)
  - Reuniones
    - Scrum diario, revisión de backlog, panificación del sprint, revisión del sprint, retrospectiva

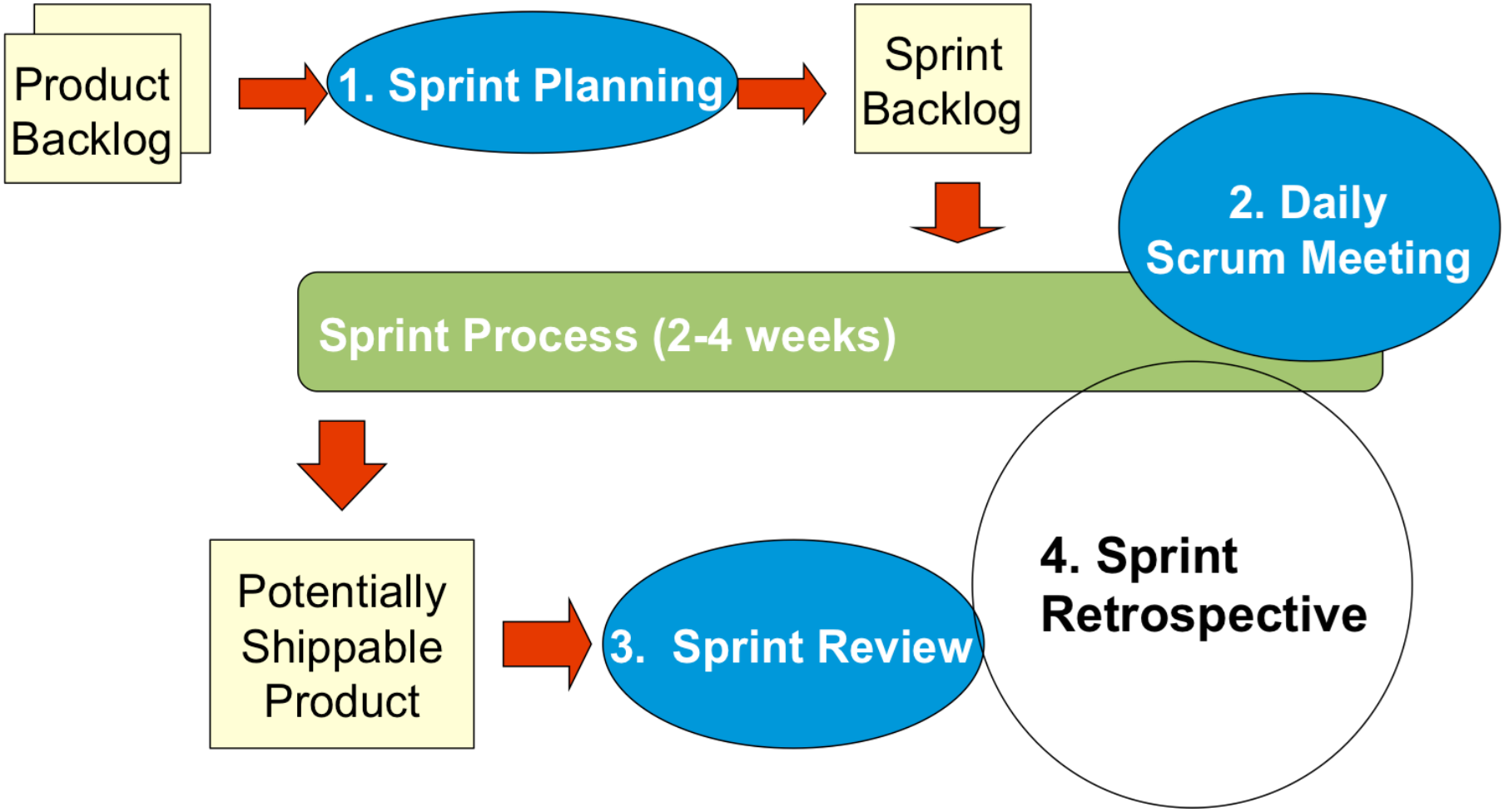
# SCRUM (2) - Roles

- *Product owner*
  - Representa la voz del cliente
  - Responsable por que el equipo aporte valor al negocio
  - Escribe las historias de usuario
  - Las prioriza y las agrega a la lista de pendientes (*backlog*)
- Equipo de desarrollo
  - Responsable por brindar un resultado de valor al final de cada *sprint*
- *Scrum master*
  - Facilitador, responsable por remover obstáculos para que el equipo genere los resultados esperados para el *sprint*
  - Asegura que el proceso *scrum* se use de forma adecuada y que el equipo mantenga el foco durante el *sprint*
  - «*Servant-leader*» del equipo

# SCRUM (3) - Artefactos

- *Backlog* del producto
- *Backlog* del *sprint*
- Incremento (lo completado desde el comienzo)
- *Burn-down chart* o gráfica del trabajo pendiente (diagrama que compara el avance respecto a lo previsto)
- Historias: forma de documentar los requisitos funcionales y eventuales requisitos no funcionales (cuentito), podría considerarse un caso de uso informal

# SCRUM (4) - Proceso



# SCRUM diario

- En el mismo lugar y hora, comienza en hora.
- Es abierto a roles auxiliares, pero normalmente solo hablan los roles principales.
- Duración fija de 15 minutos, normalmente de pie.
- Cada integrante informa 3 cuestiones:
  - Qué hice ayer, qué pienso hacer hoy, impedimentos o problemas.
- Es responsabilidad del *scrum master* resolver los impedimentos o problemas (en general por fuera de la reunión).

# Revisión y ajuste del backlog

- Estimar esfuerzo/tamaño de cada historia.
- Refinar los criterios de aceptación para cada una.
- Descomponer historias en historias más pequeñas.
- Estas reuniones
  - No pueden durar más de una hora.
  - No incluye la descomposición de una historia en tareas.
  - El equipo decide cuántas reuniones de este tipo precisa por semana.

# Planificación del *sprint*

- Al comienzo de cada *sprint.*
- Elegir qué trabajo será realizado.
- Preparar el ***sprint backlog.***
- 8 horas como máximo
  - 4 h *product owner* + equipo para priorizar el *backlog*
  - 4 h del equipo para desarrollar el plan para el *sprint*

# Revisión del *sprint*

- Revisar el trabajo completado y no completado
- Presentar el trabajo completado a los *stakeholders* (demo)
- 4 h como máximo

# Retrospectiva del *sprint*

- Todos los miembros del equipo reflexionan acerca del *sprint* pasado
- Mejora continua del proceso:
  - ¿Qué funcionó bien?
  - ¿Qué se podría mejorar para el próximo *sprint*?
  - 3 h como máximo

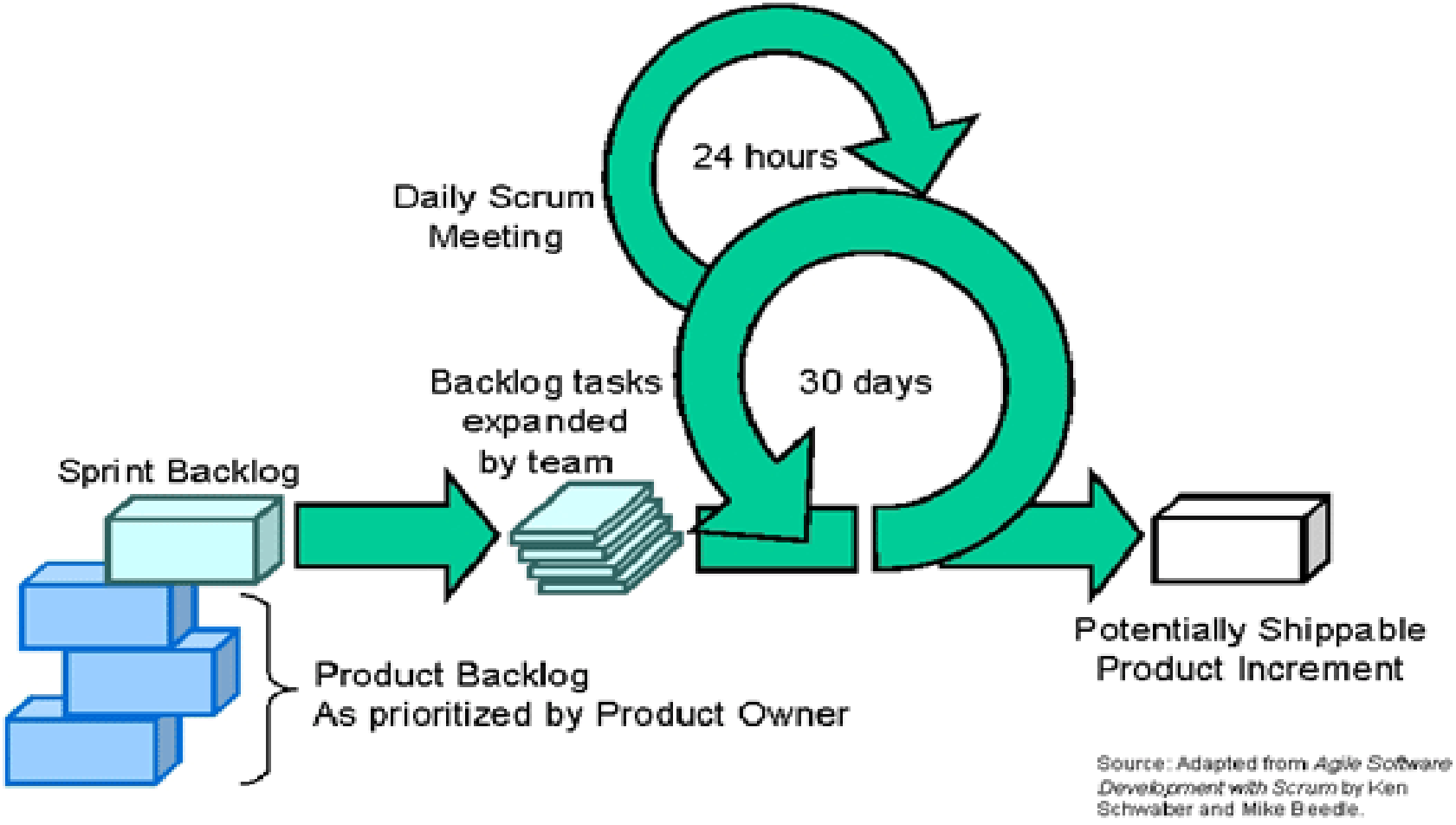
# *Backlog* del producto

- Conjunto ordenado de requisitos pendientes para un producto
- El *product owner* es el responsable del *backlog* y establece el orden considerando:
  - Riesgo, valor para el negocio, dependencias, fecha en la que es requerido, esfuerzo estimado para completarlo, etc.
- El valor para el negocio es estimado por el *product owner*

# *Backlog* del *sprint*

- Conjunto de requisitos que el equipo debe atacar en el *sprint*.
- Es propiedad del equipo de desarrollo.
- El equipo va tomando las historias que están en el tope del *product backlog* mientras piense que puede hacerlas.
- El equipo detalla para cada historia las tareas necesarias con duración de entre 4 a 16 h.
- Las tareas no se asignan, los integrantes del equipo las van tomando en el scrum diario de acuerdo a las prioridades y a las habilidades de cada miembro.
- Esto promueve la autoorganización del equipo.

# SCRUM - Resumen



# Factores relevantes para el grado de formalidad/agilidad conveniente

- **Características del proyecto**
  - Riesgos (depende de los riesgos)
    - Requisitos oscuros o cambiantes (>, > agilidad)
    - Impacto fallas (>, > formalidad)
  - Relación con otros proyectos ( >, > formalidad)
- **Características de la aplicación**
  - Criticidad (>, > formalidad)
- **Características del equipo de trabajo**
  - Competente ( >, > agilidad)
  - Tamaño (>9..., > formalidad)
- **Características del cliente**
  - Cultura, capacidades, posibilidad de involucramiento
- **Características de la relación con el cliente**
  - Desarrollo interno / Contratación externa (+confianza, > agilidad)

# Próxima clase: ingeniería de requisitos