Mejores Prácticas de la Ingeniería de Software

Objetivos

- ✓ Identificar actividades para la comprensión y resolución de problemas de la ingeniería de software.
- ✓ Explicar las seis mejores prácticas.
- ✓ Presentar al «Rational Unified Process» (RUP) en el contexto de las seis mejores prácticas.

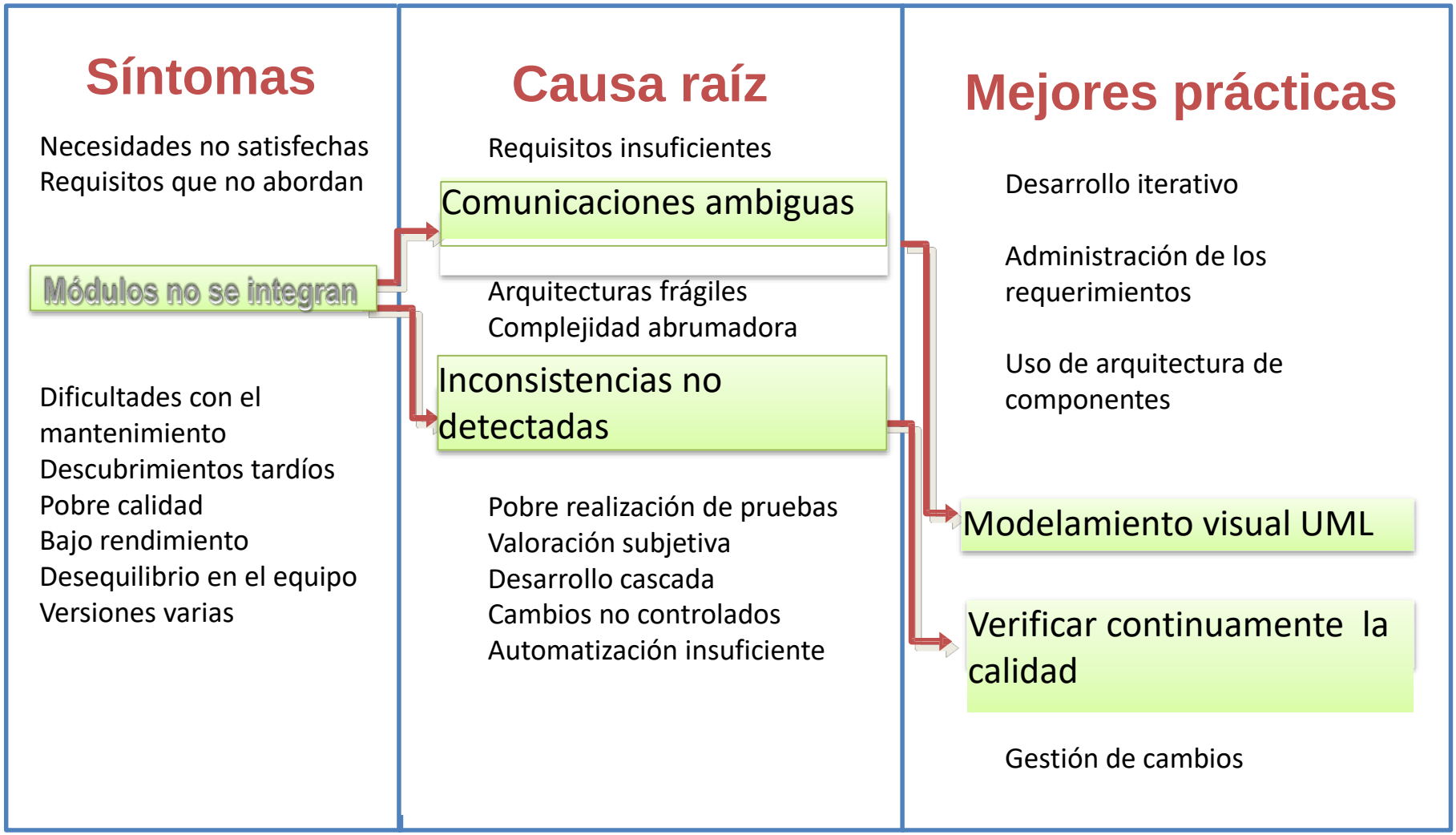
Contenido

- Los problemas de desarrollo de software
- Las seis mejores prácticas
- RUP en el contexto de las seis mejores prácticas.

Los síntomas de los problemas de desarrollo de software

- ✓ Necesidades de los usuario del negocio no satisfechas
- ✓ Requisitos no abordan
- ✓ Los módulos no se integran
- ✓ Dificultades con el mantenimiento
- ✓ Descubrimiento tardío de fallas
- ✓ La mala calidad de la experiencia del usuario final
- ✓ Bajo rendimiento, baja carga
- ✓ Ningún esfuerzo coordinado en los equipos
- ✓ Varias versiones.

Trazabilidad: Síntomas - Causa Ráiz



Contenido

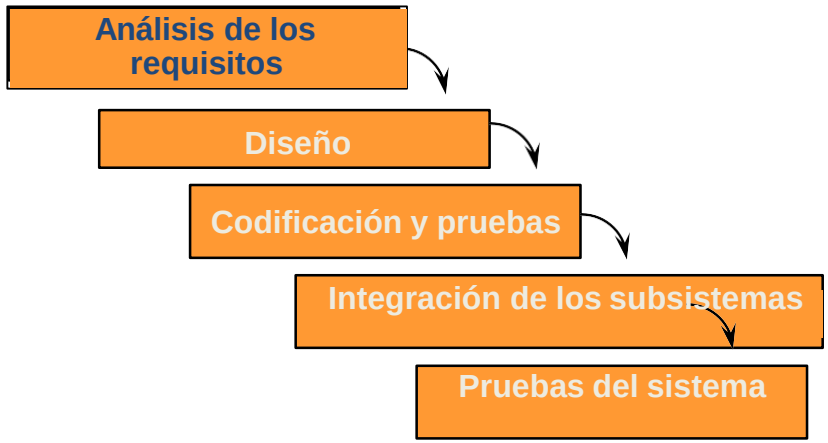
- Los problemas de desarrollo de software
- Las seis mejores prácticas
- RUP en el contexto de las seis mejores prácticas

Práctica 1: Desarrollo iterativo

Desarrollar iterativamente es una técnica que se utiliza para ofrecer la funcionalidad de un sistema en una serie sucesiva de liberaciones. Cada versión se desarrolla en un período de tiempo específico, fijo llamado una iteración.
Cada iteración se centró en definir, analizar, diseñar, construir y probar una serie de requisitos.

Características del desarrollo en cascada

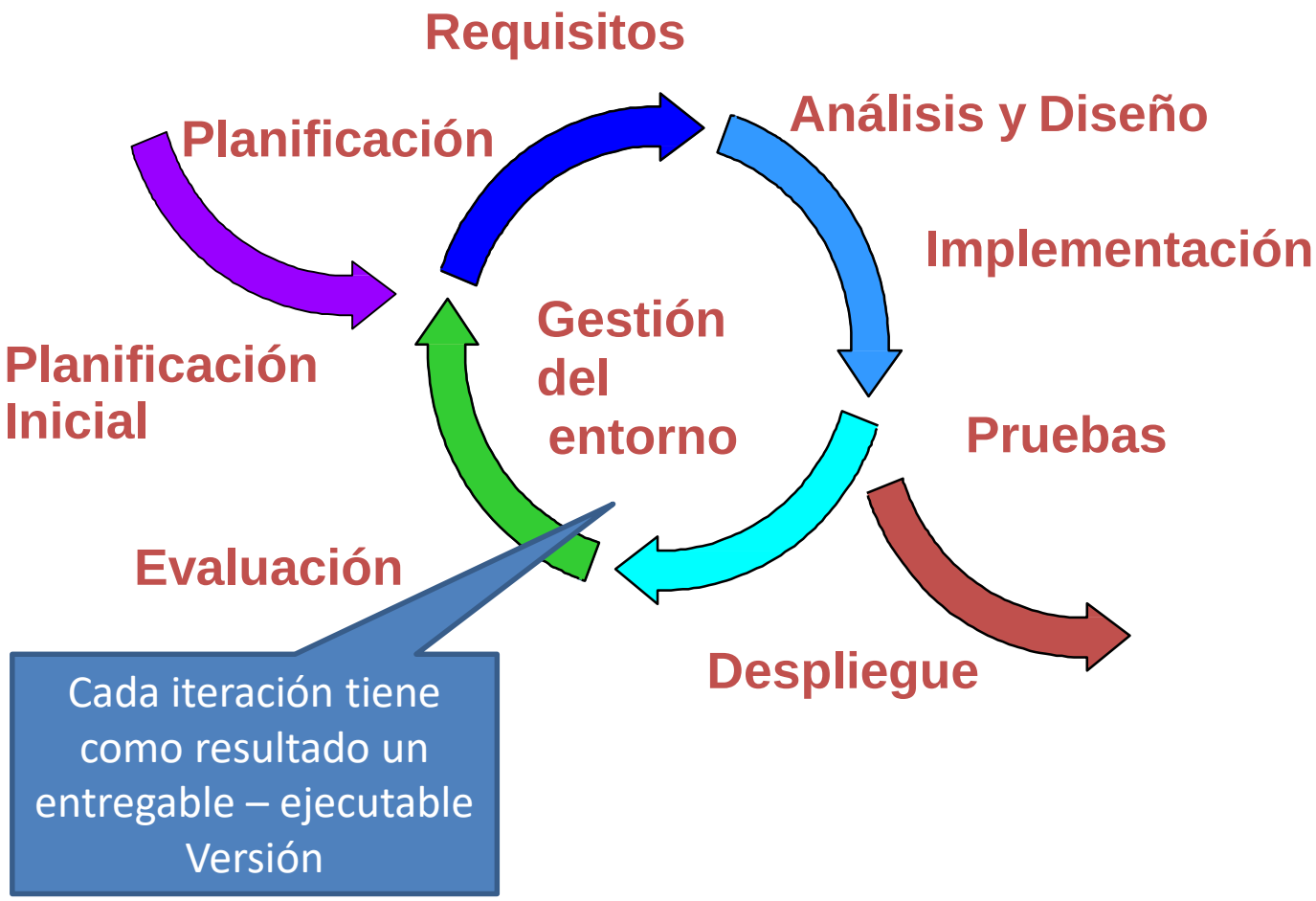
Proceso en cascada



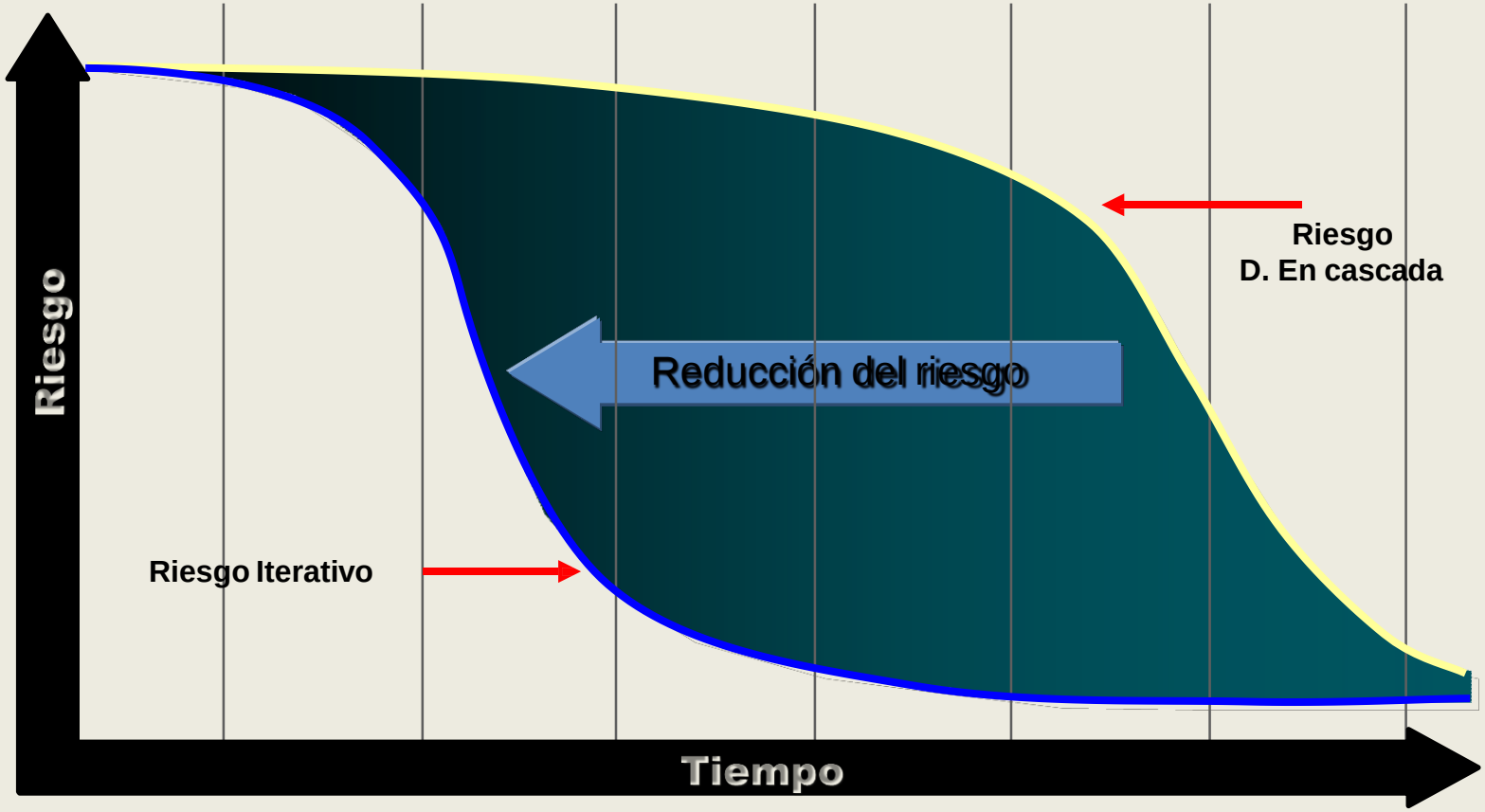
El desarrollo de la cascada es conceptualmente sencillo, ya que produce una sola entrega. El problema fundamental de este enfoque es que asume un riesgo muy grande en el tiempo, resultando muy costosa para deshacer los errores de las fases anteriores.
Un diseño inicial puede ser errónea con respecto a sus requisitos clave, y el descubrimiento tardío de defectos de diseño puede dar lugar a excesos en costos y / o cancelación de proyectos. El enfoque en cascada tiende a enmascarar los verdaderos riesgos de un proyecto hasta que es demasiado tarde para hacer algo significativo acerca de ellos.

- Retrasos en la resolución de riesgos críticos.
- Pobre medida de progreso de los productos de trabajo.
- Predicciones no acertadas de las fechas de salida en producción.
- Retrasos en la integración y pruebas
- Se opone a la implementación temprana.
- Frecuentemente resulta tener grandes iteraciones no planificados.

Desarrollo iterativo produce un entregable ejecutable



Perfiles del riesgo



- El desarrollo iterativo permite que se produzca la integración como la "actividad de verificación" de la fase de diseño permitiendo que los defectos de diseño se detecten y resuelvan antes de culminar el ciclo de vida.
- La integración contínua durante todo el proyecto reemplaza la integración "big bang" al final de un proyecto.
- El desarrollo iterativo también ofrece mucho mejor visión de la calidad.

Práctica 2:
Administración de los requisitos

Un informe del «Standish Group» confirma que una clara minoría de proyectos de desarrollo de software se ha completado a tiempo y dentro del presupuesto. En su informe, la tasa de éxito fue sólo del **16,2%,** mientras que los proyectos impugnados (operativos, tardíos y de exceso en presupuesto) representaron el **52,7%**. Proyectos de dudosa recuperación (cancelado) representaron el **31,1%**. Estos fracasos se atribuyen a los requerimientos incorrectos : Una mala definición desde el inicio del proyecto y una mala gestión de requisitos durante todo el ciclo de vida de desarrollo. (Fuente: *Chaos Report*, http://www.standishgroup.com)

***The Standish Group es una organización de carácter consultivo de investigación primaria que se centra en el rendimiento d e los proyectos de software.

Gestión de los requisitos

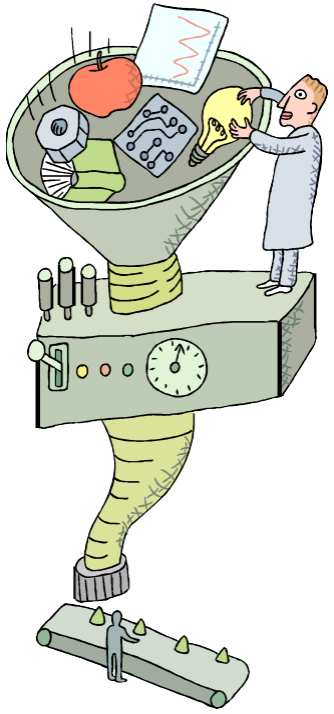
Debe asegurar que usted:

- Resolverá el problema correctamente.
- Construirá el sistema correcto.

Con la adopción de un enfoque sistémico para:

- Obtener
- Organizar
- Documentar
- Gestionar

Las necesidades cambiantes de una aplicación de software.



Aspectos de la gestión de los requisitos

- ✓ Analizar el problema
- ✓ Entender las necesidades de los usuarios
- ✓ Definir claramente el sistema
- ✓ Gestionar el alcance
- ✓ Afinar la definición del sistema
- ✓ Administrar los cambios de las necesidades

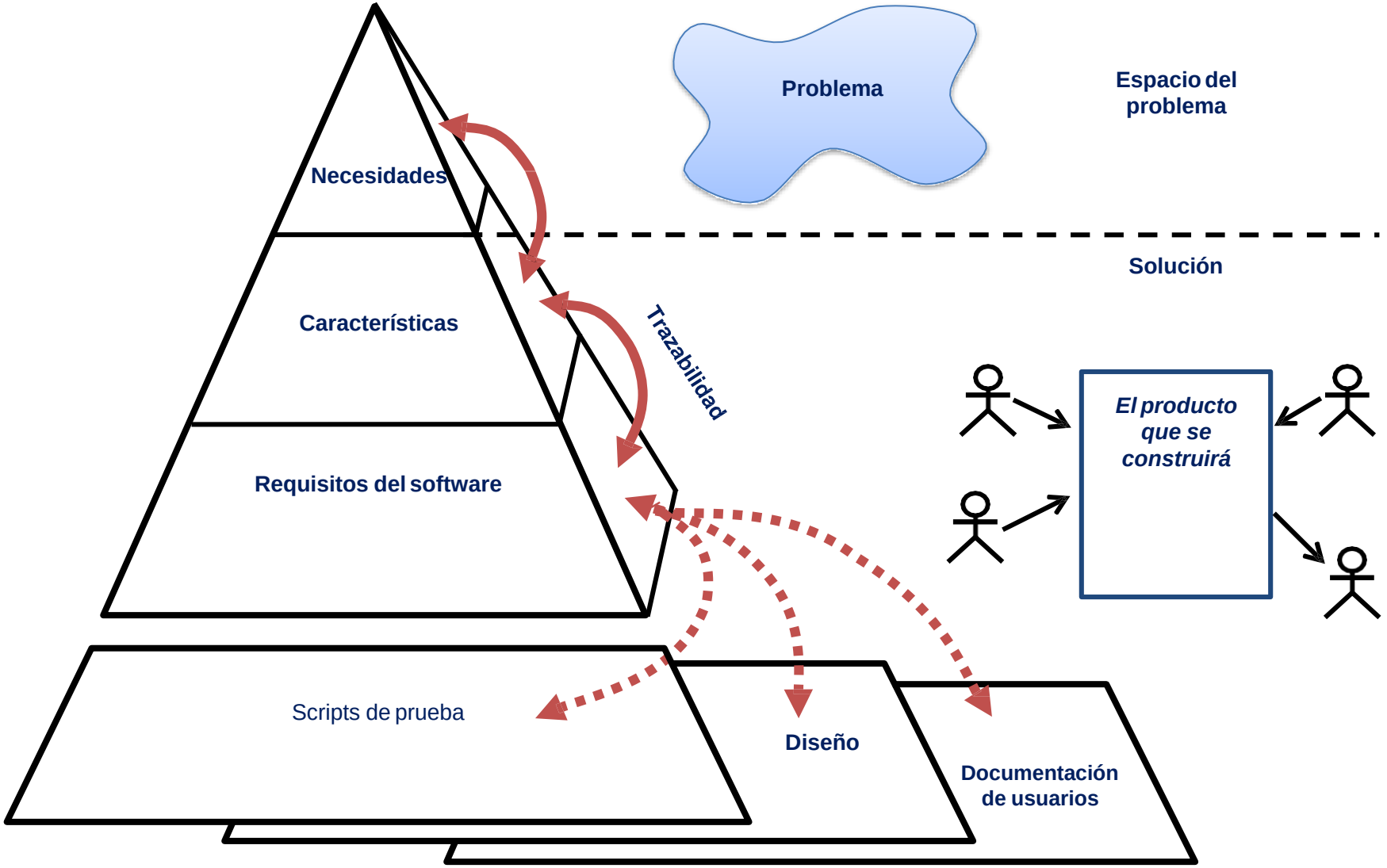
La gestión de los requisitos

La gestión de requisitos implica la traducción de las peticiones de las partes interesadas en un conjunto de necesidades de los interesados clave y funciones del sistema. Estos a su vez se detallan en las especificaciones de requisitos funcionales y no funcionales. Las especificaciones detalladas se traducen en procedimientos de prueba, diseño y documentación de usuario.

La trazabilidad nos permite:

- ✓ Evaluar el impacto de los proyectos de un cambio en un requisito.
- ✓ Evaluar el impacto de una falla de una prueba sobre los requisitos (es decir, si la prueba falla, el requisito puede no ser satisfecho).
- ✓ Administrar el alcance del proyecto.
- ✓ Verifique que todos los requisitos del sistema se cumplen por la implementación.
- ✓ Compruebe que la aplicación hace sólo lo que se pretende hacer.
- ✓ Gestionar el cambio.

Mapa del territorio



Práctica 3:
Use arquitecturas de componentes

La arquitectura de un software en el desarrollo de productos ofrece mayor retorno de la inversión con respecto a la calidad, el cronograma y los costos, según los autores de «Software Architecture in Practice" (Len Bass, Paul Clements and Rick Kazman [2012] Addison-Wesley).

Definición de un componente (software):

RUP: Una parte no trivial, casi independiente, y reemplazable de un sistema que realiza una función clara en el contexto de una arquitectura bien definida. Un componente se ajusta y proporciona la realización física de un conjunto de interfaces.

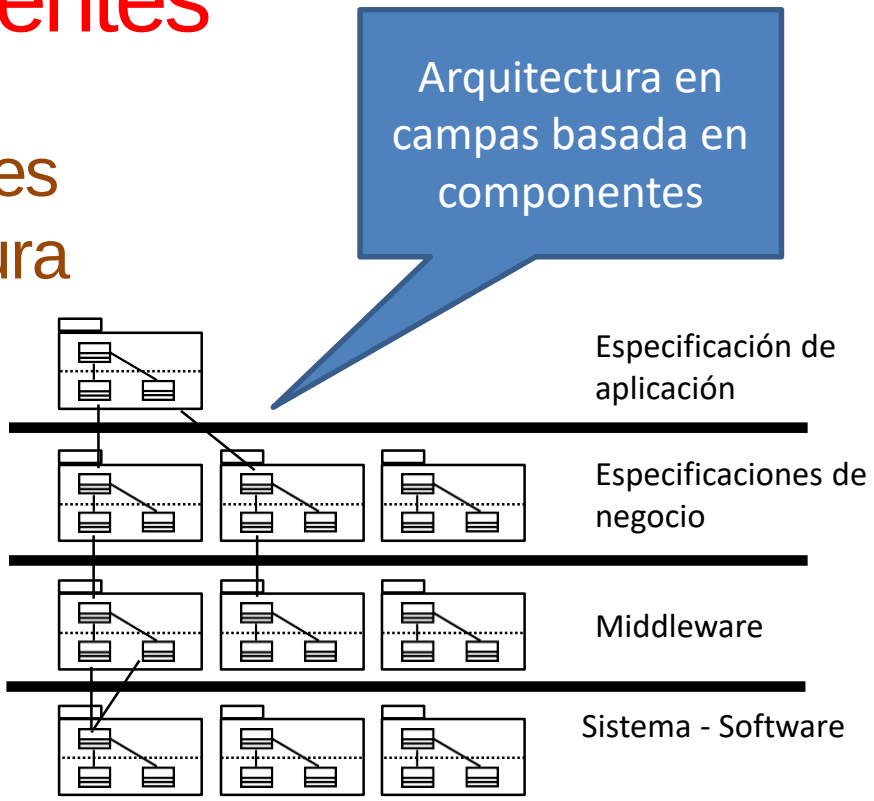
UML: Una parte física, reemplazable de un sistema que dispositivos de aplicación, y que se ajusta y proporciona la realización de un conjunto de interfaces. Un componente representa una pieza física de la implementación de un sistema, incluyendo el código de software (fuente, binario o ejecutable) o equivalentes, tales como scripts o archivos de comandos.

Arquitecturas basado en componentes : Flexibilidad

- Flexibilidad, Elásticidad:
  - Cumple con los requerimientos actuales y futuros
  - Mejora la extensibilidad
  - Permite la reutilización
  - Encapsula las dependencias del sistema
- Basado en componentes:
  - Reutilizar o personalizar componentes
  - Seleccione de componentes disponibles en el mercado
  - Evolucionar software existente de forma incremental

Propósito de una arquitectura basado en componentes

- Bases para la reutilización
  - Reutilización de componentes
  - Reutilización de la arquitectura
- Bases para la gestión de proyectos
  - Planificación
  - Contratación de personal
  - Entrega
- Control Intelectual
  - Administrar la complejidad
  - Mantener la integridad



Práctica 4:
Modelo Visual (UML)

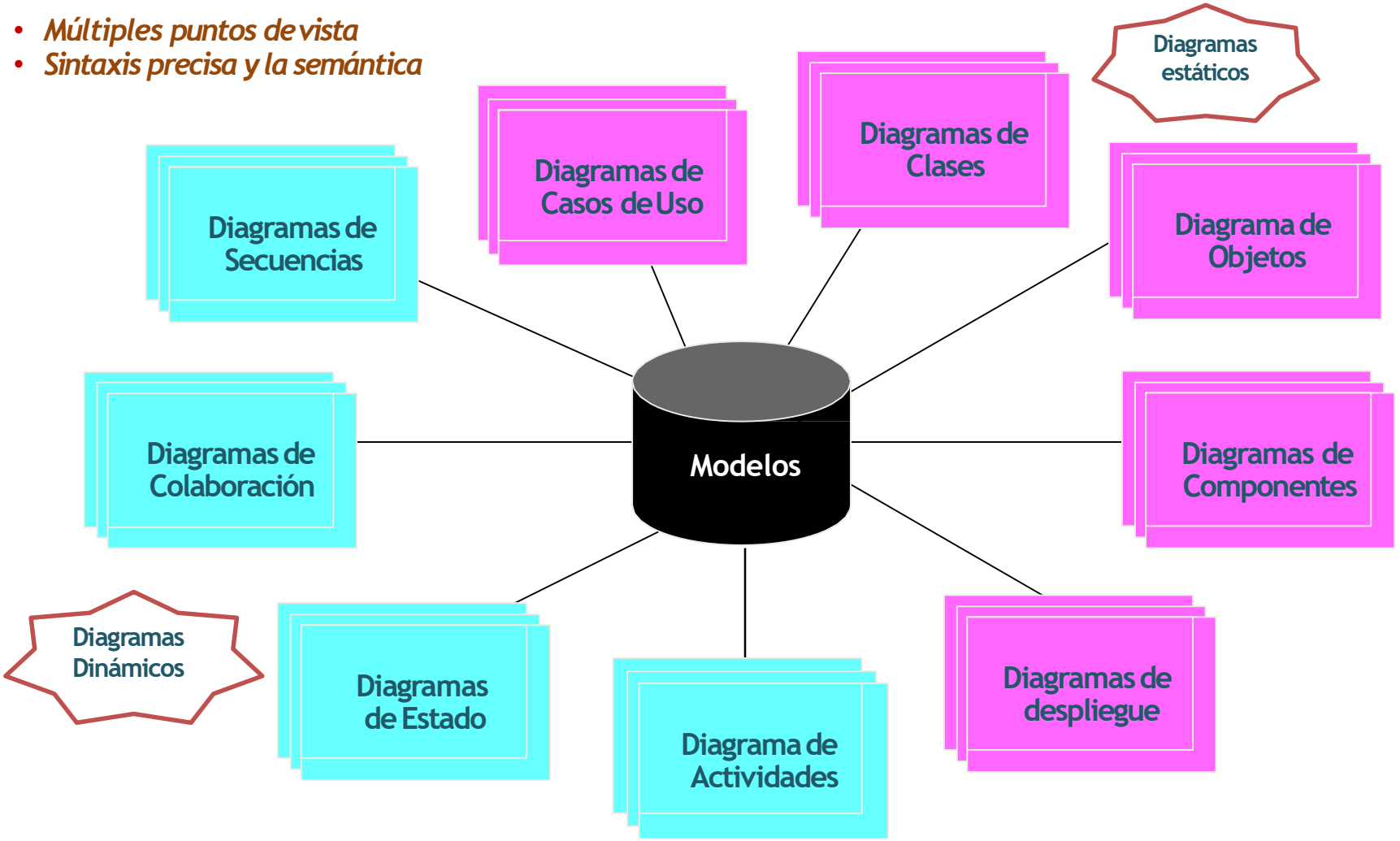
Un modelo es una simplificación de la realidad que proporciona una descripción completa de un sistema desde una perspectiva particular. Construimos modelos para que podamos entender mejor el sistema que estamos construyendo. Construimos modelos de sistemas complejos porque no podemos comprender cualquier sistema en su totalidad.

¿Por qué un modelo visual?

- ✓ Captura la estructura y el comportamiento
- ✓ Muestra cómo encaja los elementos de un sistema
- ✓ Mantiene el diseño y la aplicación coherente
- ✓ Oculta o expone los detalles apropiados
- ✓ Promueve la comunicación sin ambigüedades
- ✓ El UML provee un idioma para todos los profesionales independientemente del lenguaje de programación que se pueda utilizar.

Modelado Visual con UML

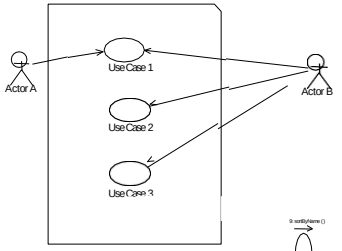
- *Múltiples puntos de vista*
- *Sintaxis precisa y la semántica*

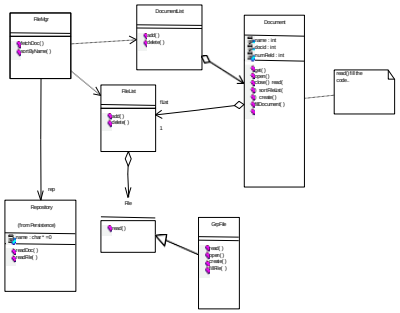


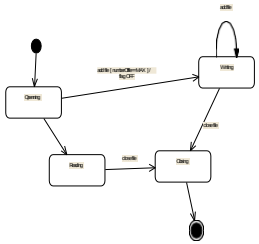
Modelo Visual

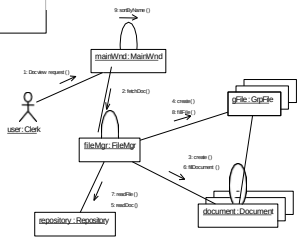
- ✓ En la construcción de un modelo visual de un sistema, se necesitan muchos esquemas diferentes para representar diferentes vistas del sistema.
- ✓ El UML proporciona una notación rica para la visualización de modelos.
- ✓ Esto incluye los siguientes diagramas clave:
  - Diagramas de casos de uso para ilustrar las interacciones del usuario con el sistema.
  - Los diagramas de clases para ilustrar estructura lógica.
  - Diagramas de objetos para ilustrar objetos y enlaces
  - Diagramas de componentes para ilustrar la estructura física del software.
  - Los diagramas de despliegue para mostrar la cartografía de software para configuraciones de hardware
  - Los diagramas de actividades para ilustrar los flujos de eventos
  - Los diagramas de estado para ilustrar el comportamiento
  - Los diagramas de interacción (es decir, la colaboración y los diagramas de secuencia) para ilustrar el comportamiento

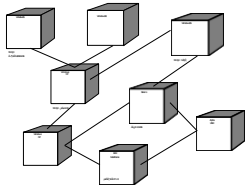
Modelamiento Visual haciendo uso de UML

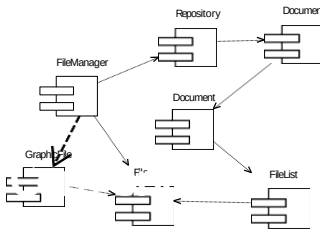


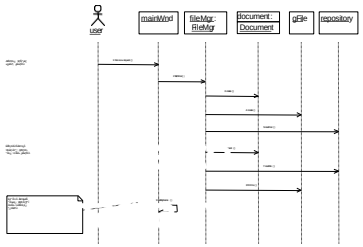




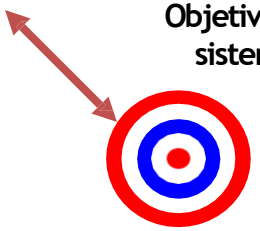








Objetivo del sistema



Ingeniería directa e inversa

Práctica 5:
Verificar la calidad continuamente

Calidad, como se usa en el RUP, se define como "La característica de haber demostrado el logro de la producción de un producto que cumple o excede los requisitos acordados, según lo medido por medidas acordadas y criterios, y es producido por un proceso acordado ".

La calidad también incluye la identificación de las medidas y criterios y la implementación de un proceso para asegurar que el producto resultante ha alcanzado el grado deseado de calidad.

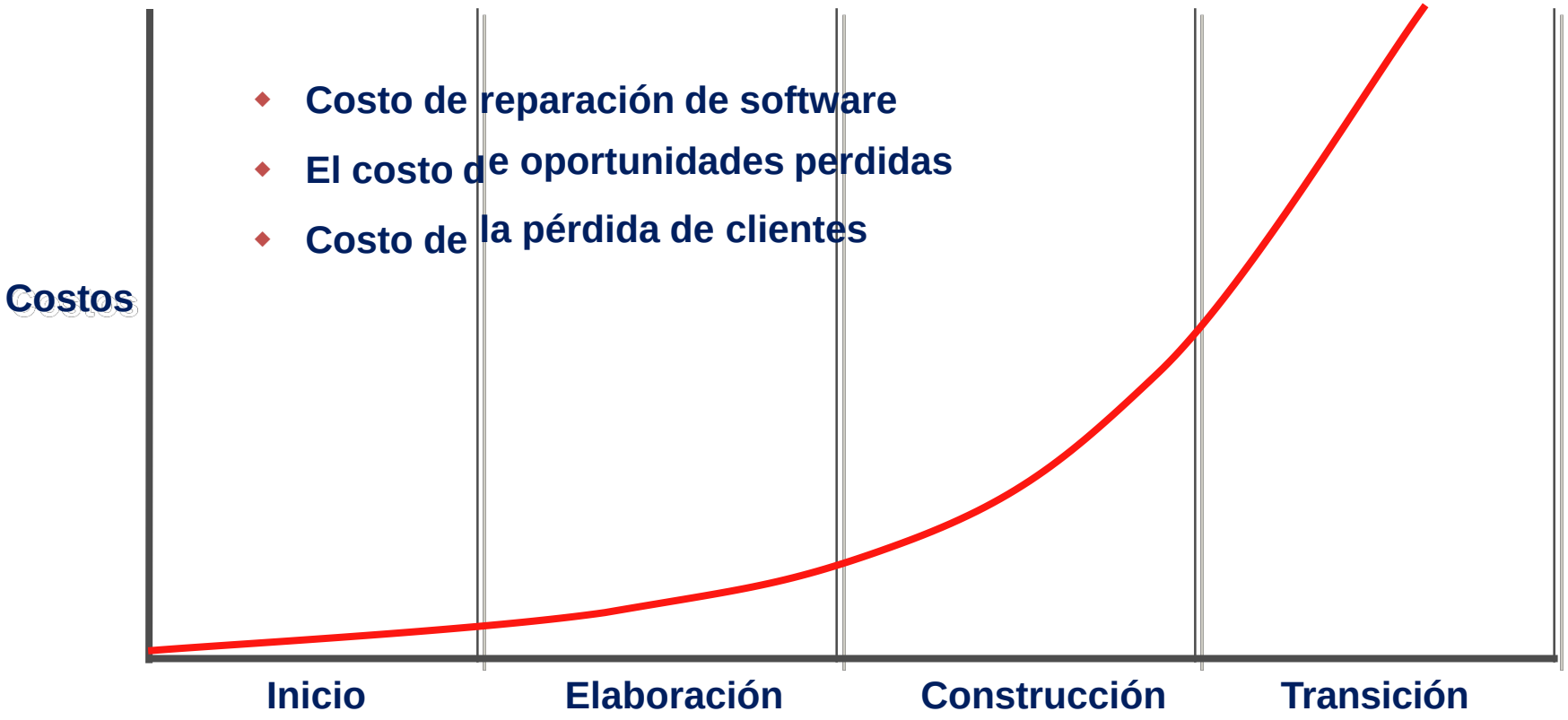
En primer lugar, es enormemente difícil probar el software. Las diferentes formas en que un determinado programa puede comportarse son casi infinitas.

En segundo lugar, la prueba se realiza normalmente sin una metodología clara y sin el apoyo de automatización o herramienta requerida.

Aunque la complejidad del software hace que la prueba completa una meta imposible, una metodología y uso de herramientas bien concebidas puede mejorar en gran medida la productividad y la eficacia de las pruebas de software

Continuamente verifique la calidad de su software

Los problemas de software son de 100 a 1000 veces más costosos si se encuentran y repararan después de la implementación



Comprobación de dimensiones de la calidad

Usabilidad

- Aplicación de pruebas desde la perspectiva de conveniencia para el usuario final.

Funcionalidad

- Pruebe el funcionamiento preciso de cada escenario de uso.

Confiabilidad

- Pruebe que la aplicación se comporta de forma coherente y previsible.

Soportabilidad

- Ponga a prueba la capacidad de mantener y apoyar la aplicación en condiciones de uso de producción.

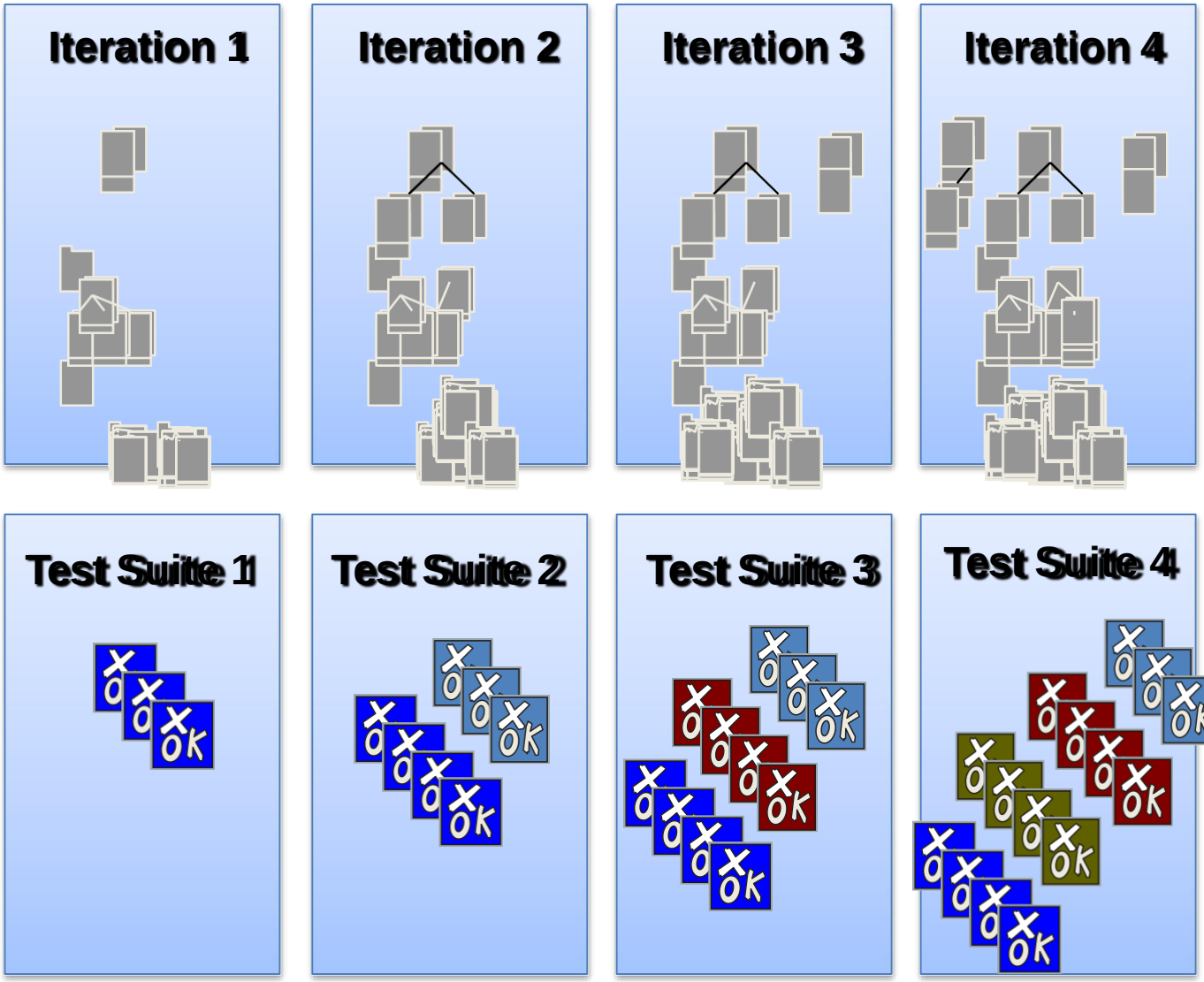
Rendimiento

- Ponga a prueba la respuesta en línea bajo cargas promedio y pico.

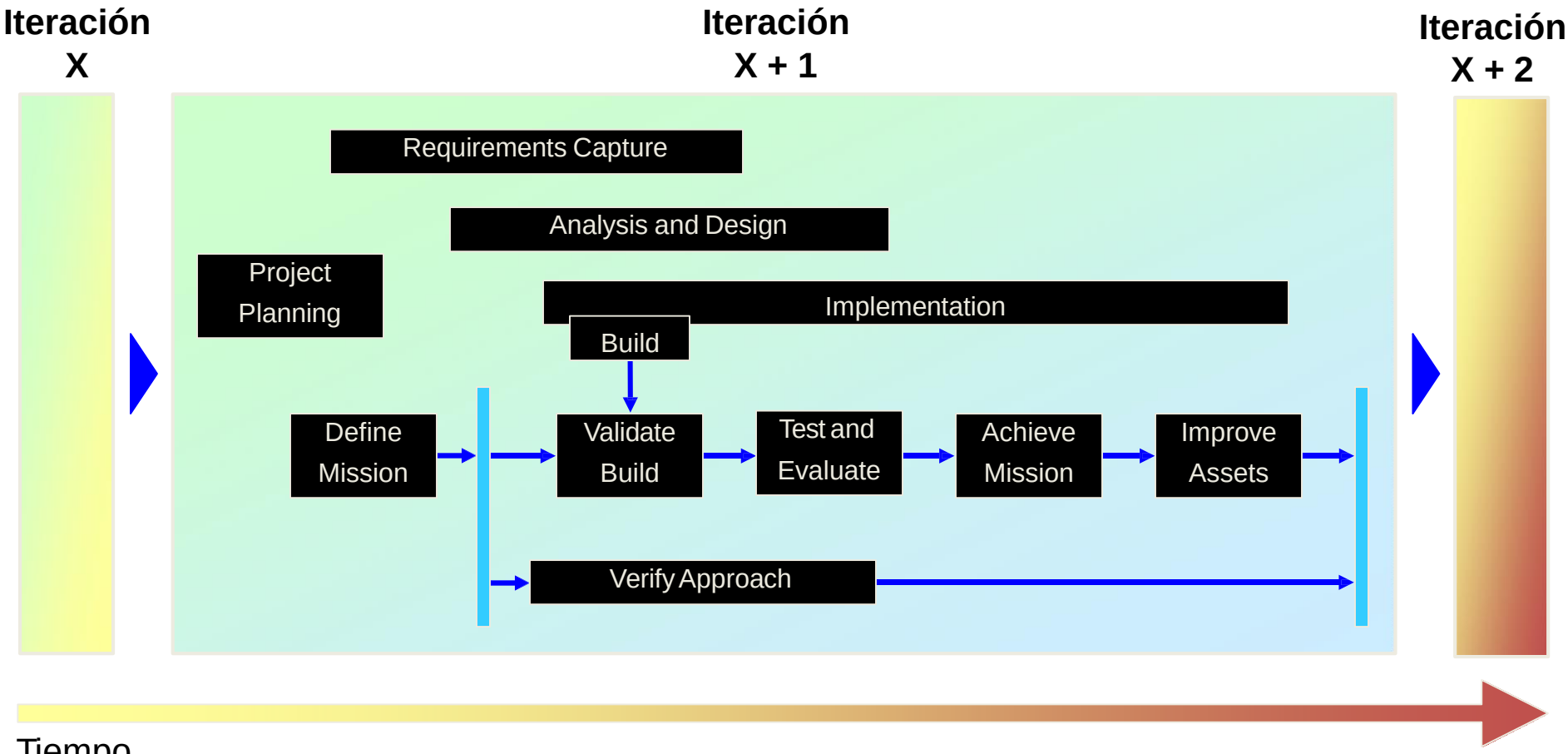
Probar en cada iteración

Modelamiento UML e implementación

Pruebas



Prueba dentro del ciclo de vida de desarrollo de productos

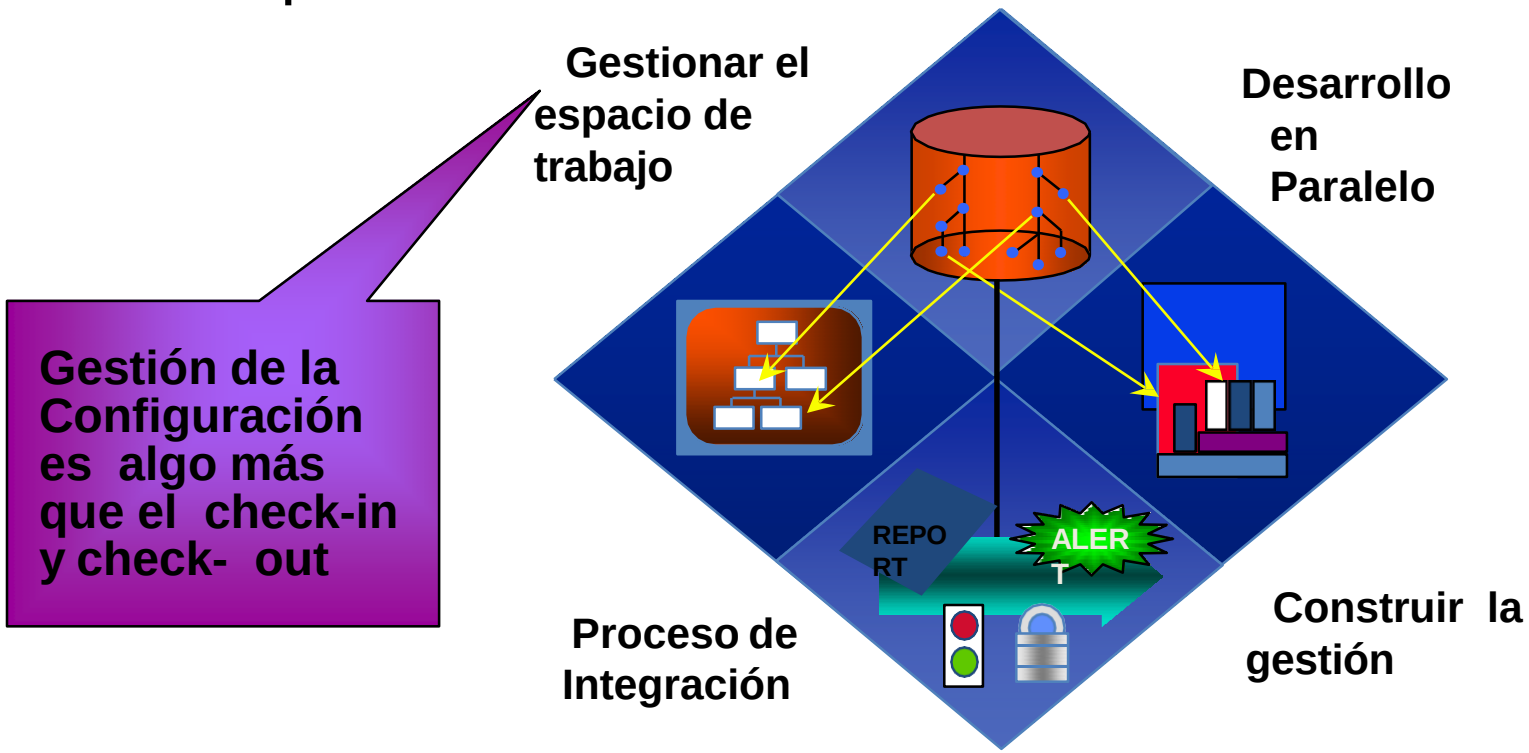


Práctica 6:
Gestión de cambios

No se puede detener la variedad de solicitudes de cambio en proyecto de software, sin embargo se debe controlar cómo y cuándo se introducen los cambios en los artefactos del proyecto, y que hace que se introduzcan esos cambios.
Unified Change Management (UCM) es el enfoque de Rational Software para la gestión del cambio en el desarrollo de sistemas de software, de los requisitos para liberar.

¿Qué es lo que quieres controlar?

- Asegurar espacios de trabajo para cada desarrollador.
- Automatizar laa gestión y la integración.
- Desarrollo paralelo.



Aspectos de la gestión de cambios de un sistema

- ✓ Solicitud de cambios
- ✓ Estado de la configuración de informes
- ✓ Gestión de la configuración
- ✓ Seguimiento de los cambios
- ✓ Control de las versiones
- ✓ Fabricación de software

Unified Change Management (UCM)
Gestión del cambio unificado

UCM incluye:

Gestión de todo el ciclo de vida
- Sistema
- Gestión de proyectos

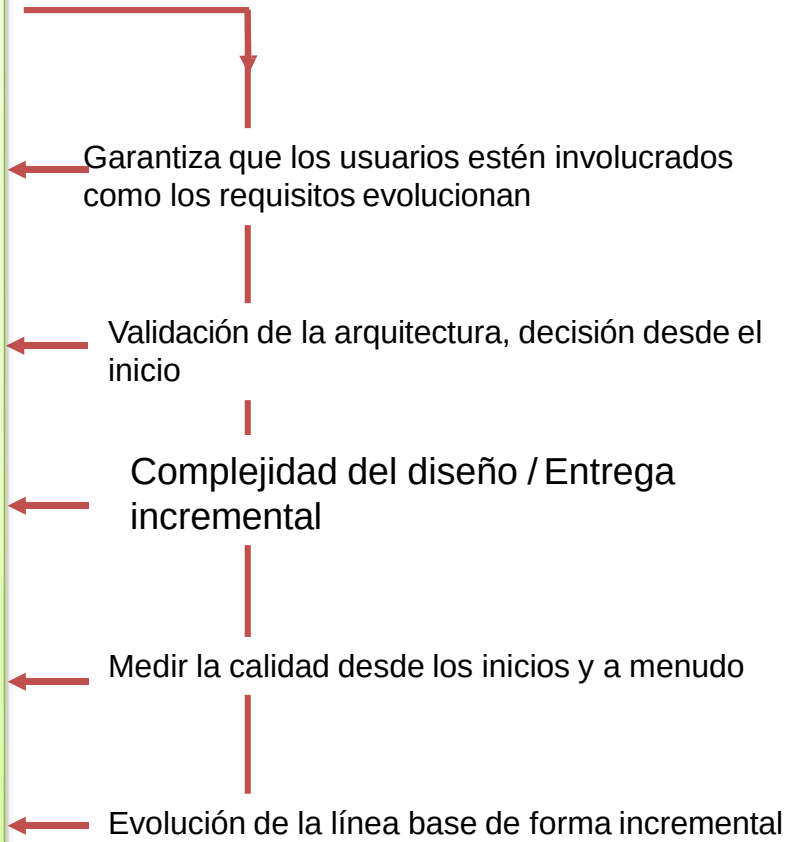
Gestión por actividades;
- Tareas
- Defectos
- Mejoras

Seguimiento de los progresos
- Gráficos
- Informes

Las mejores prácticas se refuerzan mutuamente

Mejores practicas

- Desarrollo iterativo
- Gestión de los requisitos
- Use arquitectura de componentes
- Modelo Visual (UML)
- Verifique continuamente la calidad
- Gestión de cambios



Contenido

- Los problemas de desarrollo de software
- Las seis mejores prácticas
- RUP en el contexto de las seis mejores prácticas

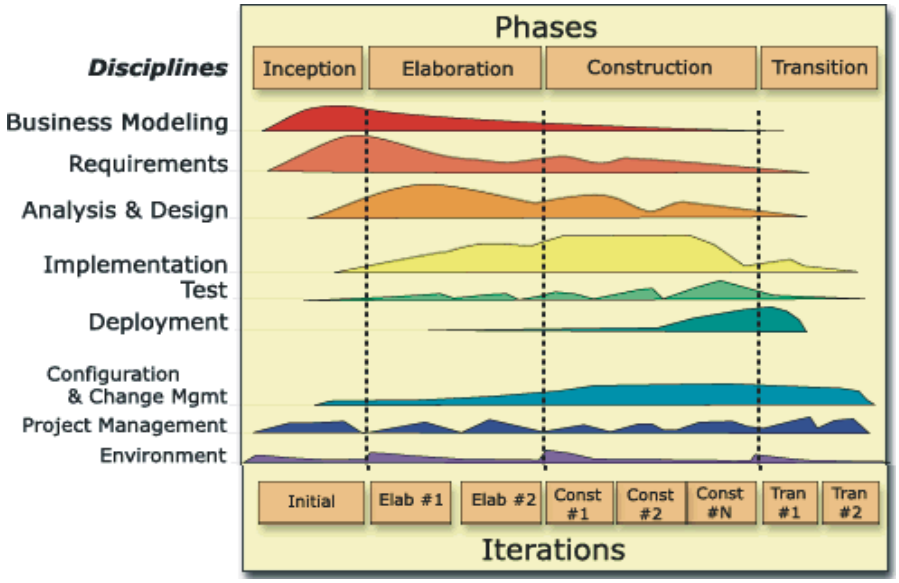
Rational Unified Process
Implementa mejores practicas

Mejores prácticas

- Desarrollo iterativo
- Gestión de los requisitos
- Use arquitectura de componentes
- Modelo Visual (UML)
- Verifique continuamente la calidad
- Gestión de cambios

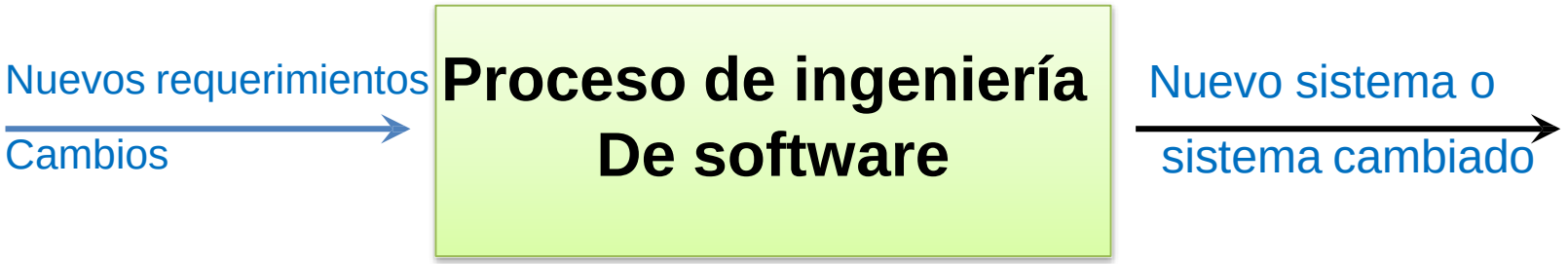
El logro de las mejores prácticas

- ✓ Enfoque iterativo
- ✓ Orientación para las actividades y artefactos
- ✓ Enfoque de procesos en la arquitectura
- ✓ Los casos de uso que impulsan el diseño e implementación
- ✓ Los modelos que resumen el sistema.

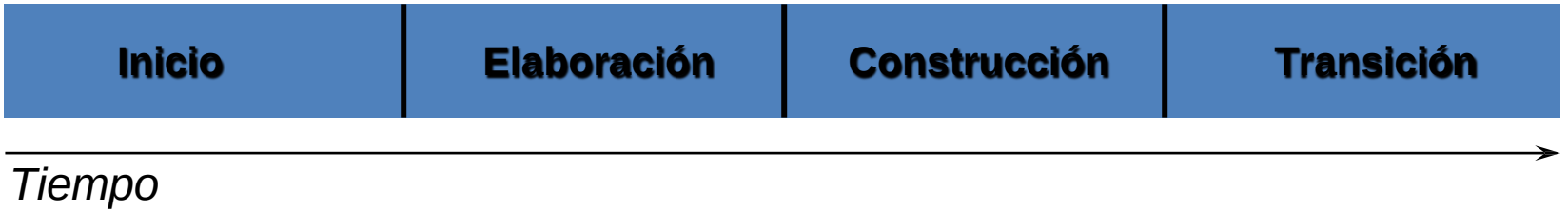


Un equipo basado en la definición del proceso

Un proceso define **quién** esta haciendo **Qué, Cuándo,** y **Cómo** , con el fin de llegar a un objetivo.



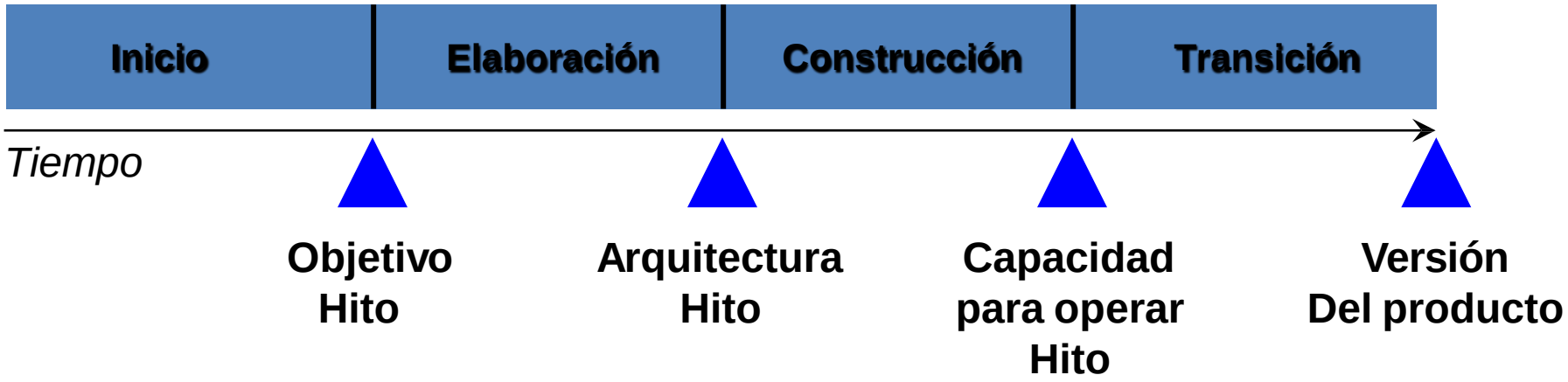
Estructura de Procesos - Fases del ciclo de vida



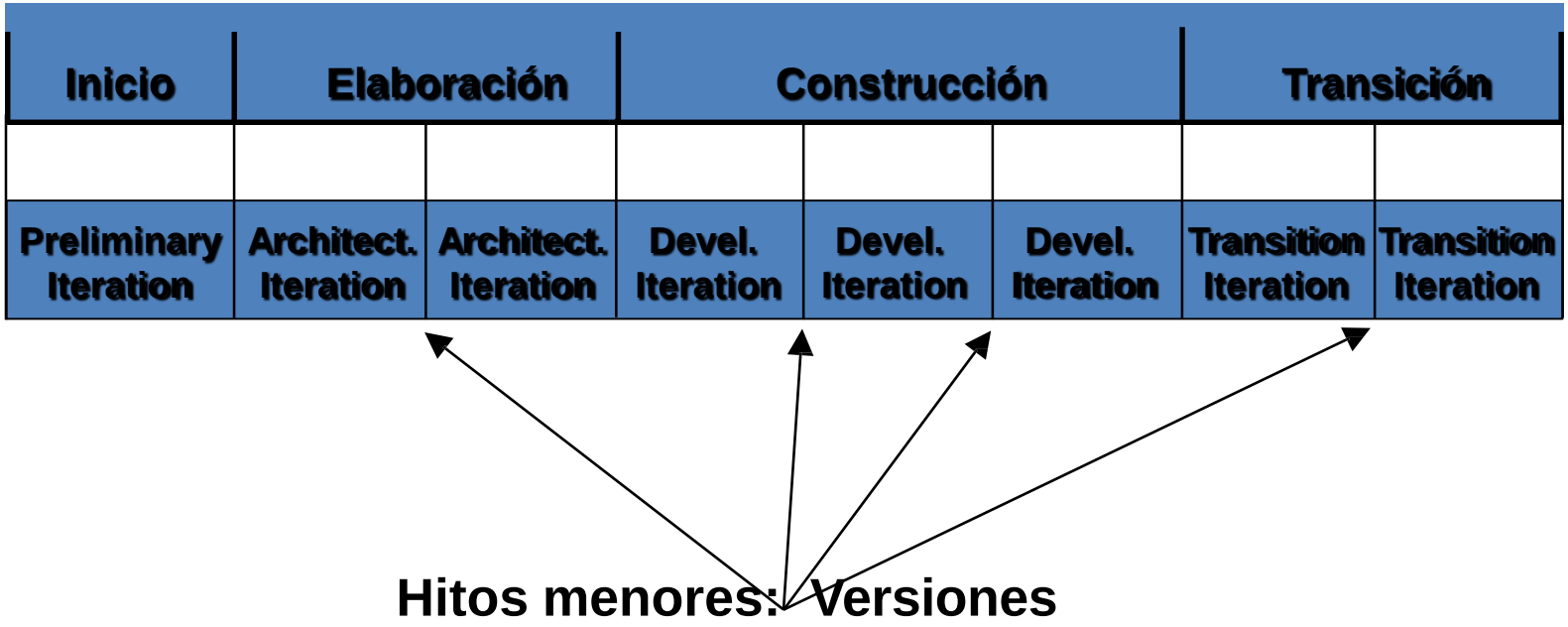
Rational Unified Process tiene 4 fases:

- **Inicio** - Definir el alcance del proyecto
- **Elaboración** – Plan del proyecto, especifique las características y arquitectura de referencia.
- **Construcción** – Construcción del producto
- **Transición** - Transición del producto en la comunidad de usuarios finales

Límites de cada fase
Marcar hitos principales

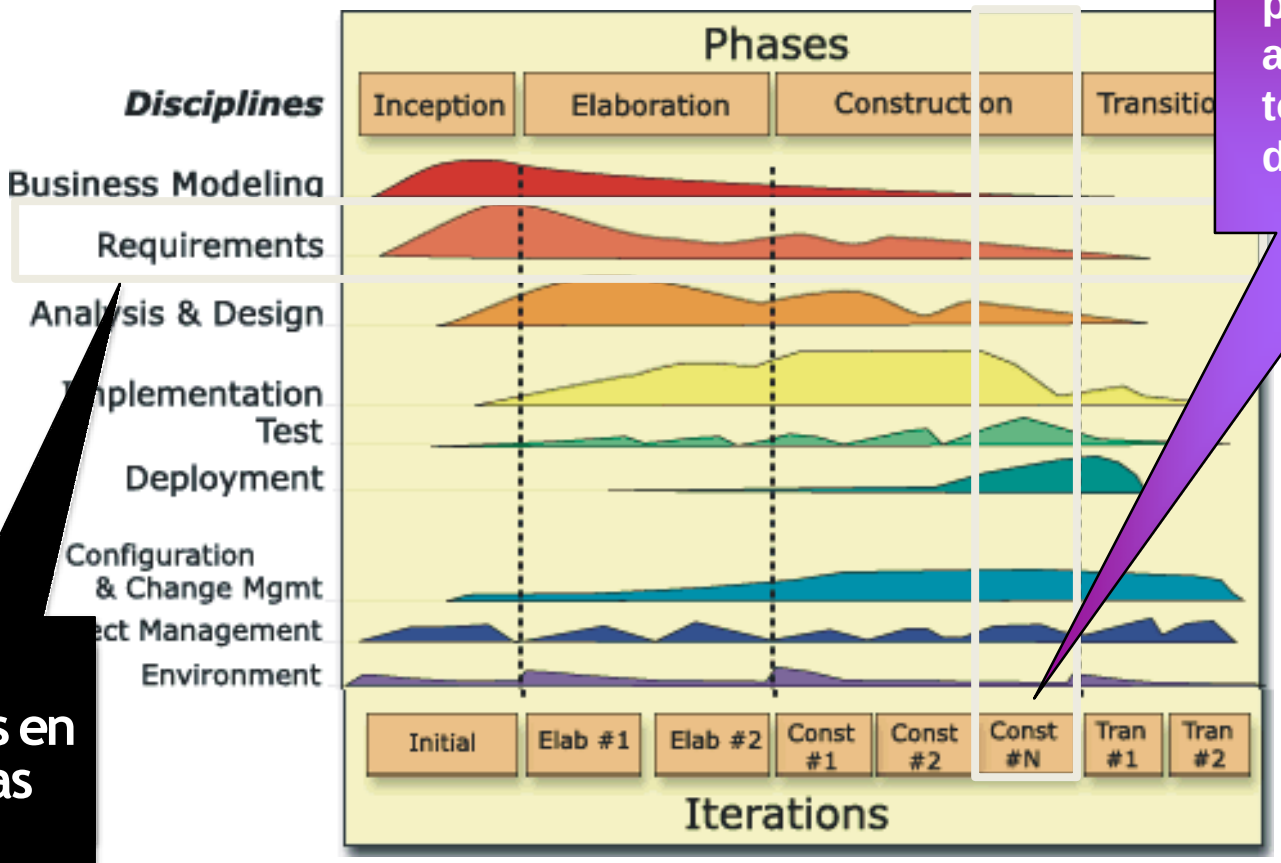


Iteraciones y fases



Una iteración es una secuencia distinta de las actividades sobre la base de un plan y criterios de evaluación establecidos, lo que resulta un entregable ejecutable (interna o externa).

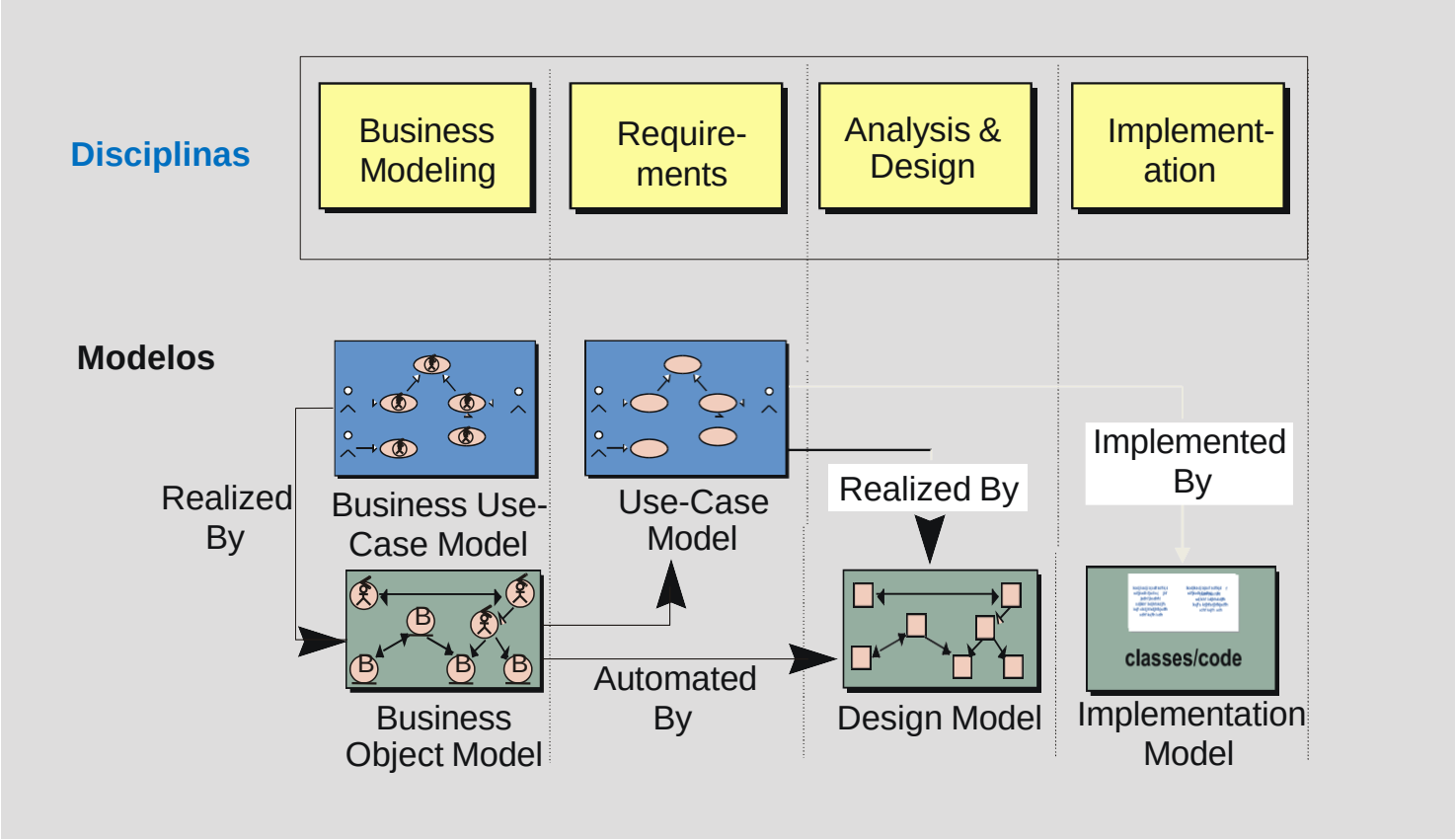
Realizar todas las actividades sucesivamente : el enfoque iterativo



En una iteración, se puede caminar a través de todas las disciplinas.

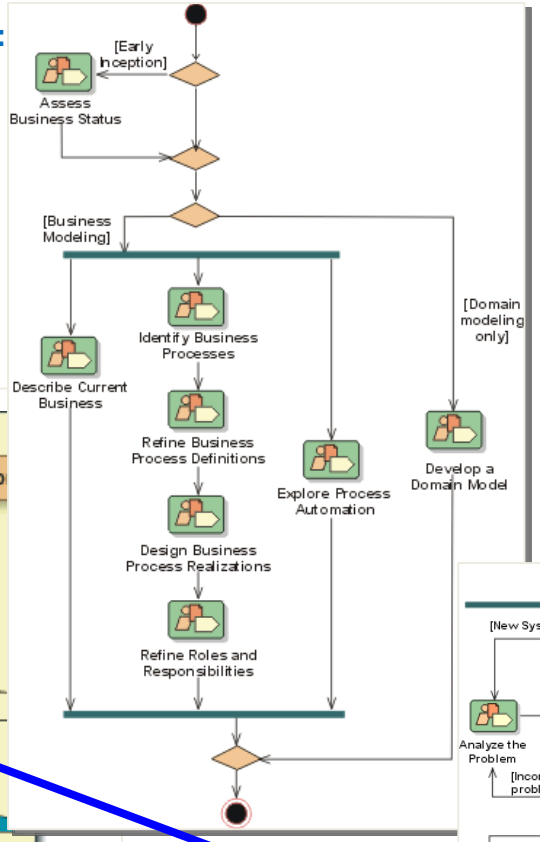
Actividades en grupo: todas las fases

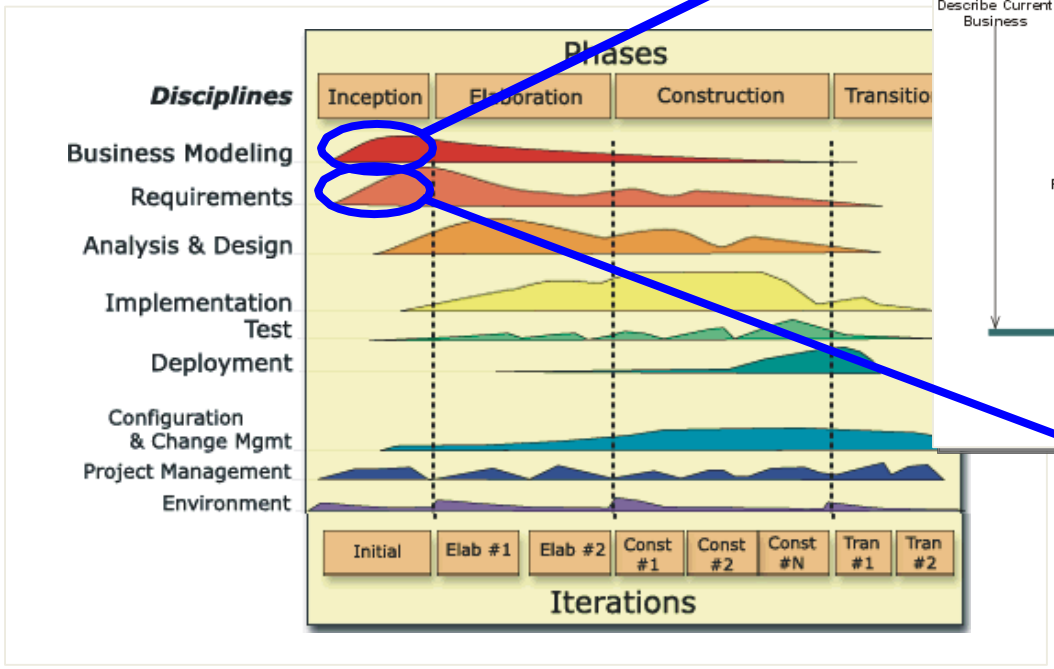
Disciplina para producir modelos

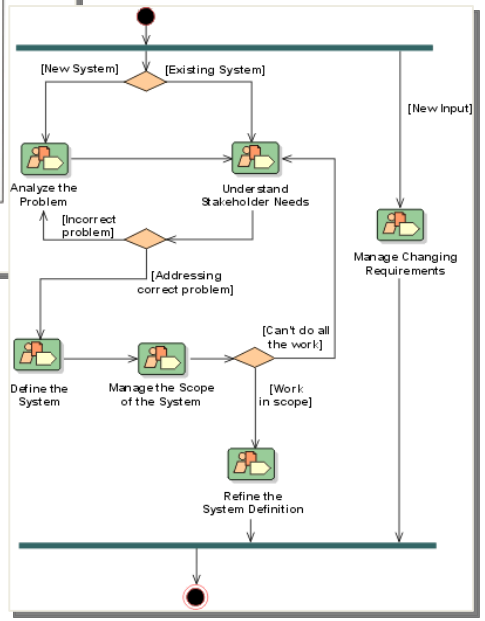


Guía de desarrollo iterativo

Modelo del negocio: Flujo de trabajo

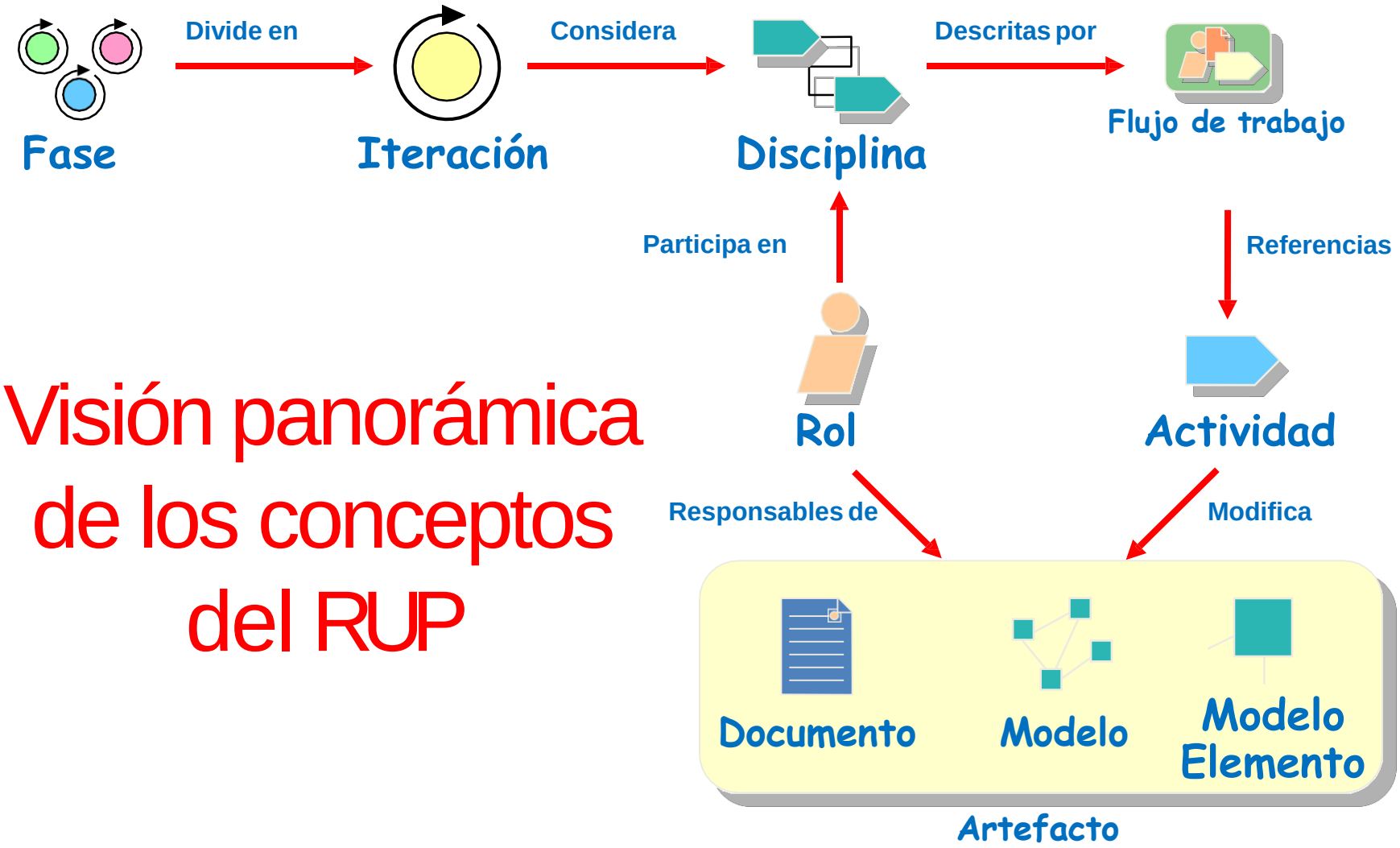






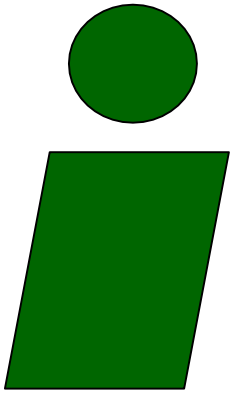
Requerimientos del flujo de trabajo

Visión panorámica de los conceptos del RUP



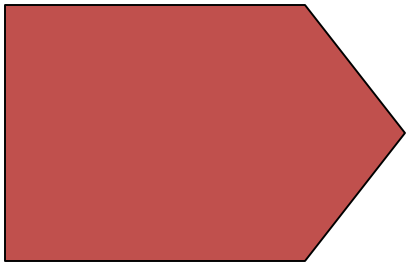
Definiciones en RUP…(1)

ROL



- Un rol define las responsabilidades y el comportamiento de un individuo.
- Es como un "sombrero" que la persona usa durante el proyecto:
  - ✓ Una persona puede tener varios sombreros.
  - ✓ Es el rol que desempeña en un momento dado.

Definiciones en RUP…(2)

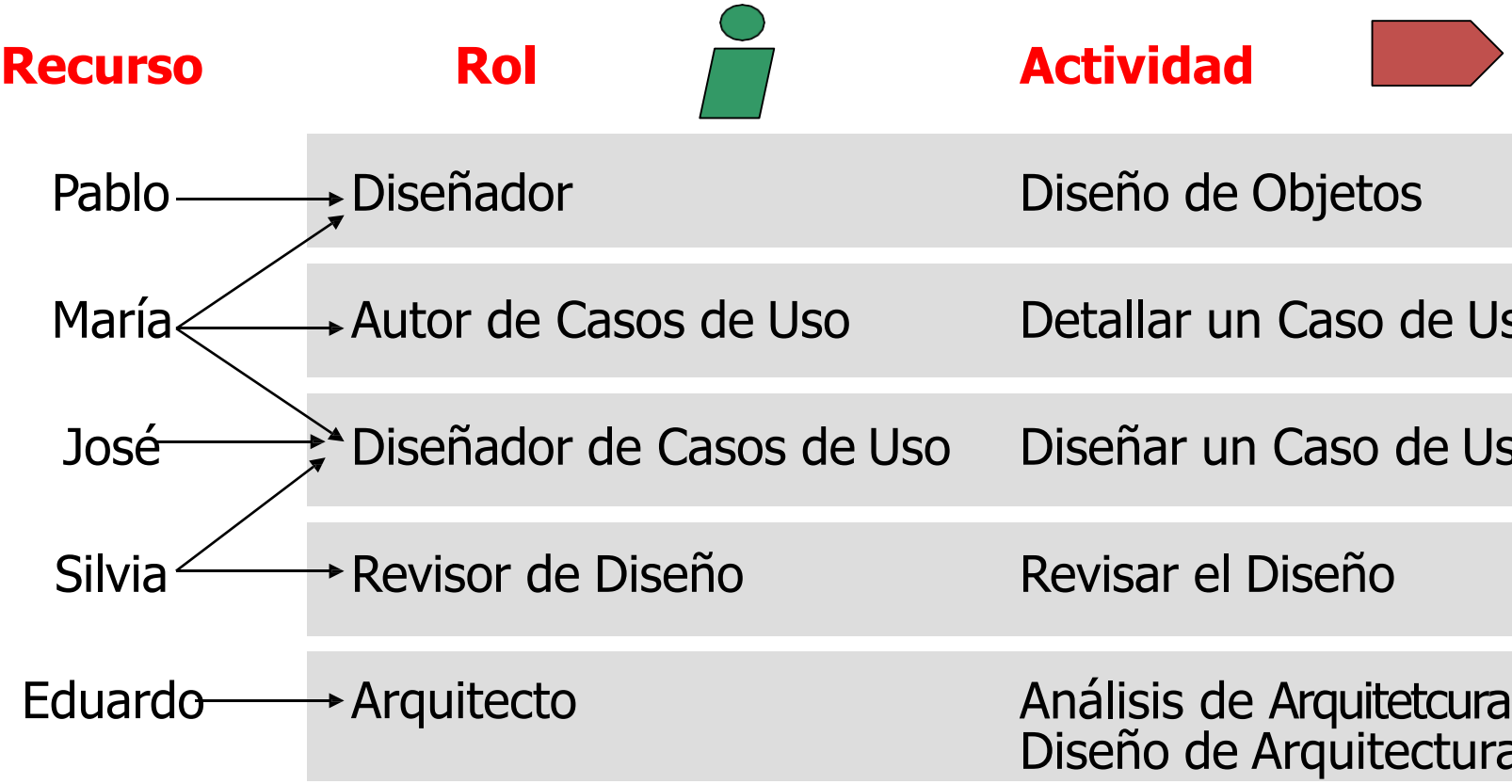


ACTIVIDAD:

- Es una unidad de trabajo que se asigna a un rol. Ejemplos: Crear o modificar una clase.
- Una actividad lleva entre un par de horas y un par de días, involucra un solo rol y un número pequeño de artefactos.
- Las actividades se consideran en la planificación y evaluación del progreso de un proyecto.

Asignación de actividades

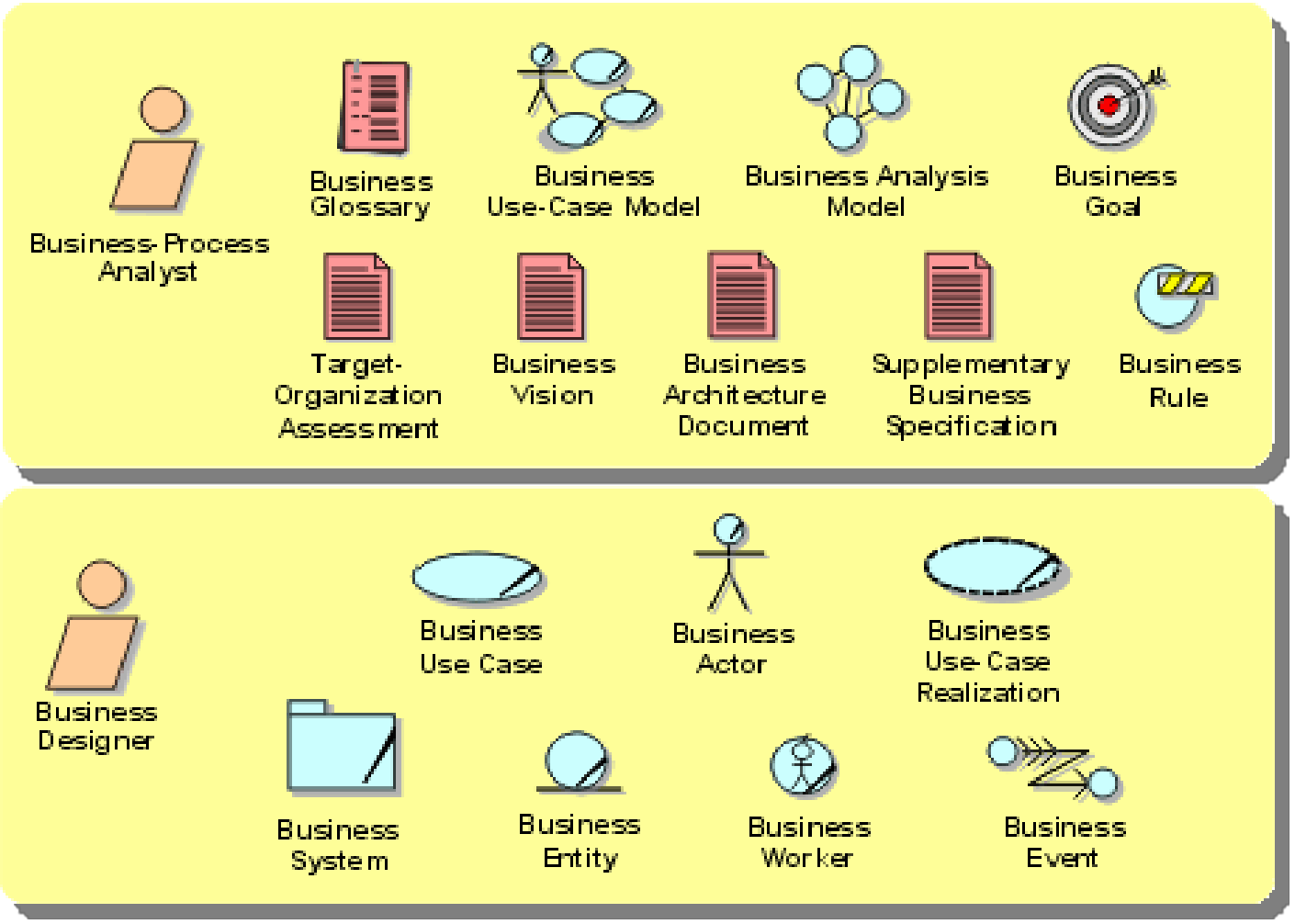
¿Cómo realizo una asignación de actividades?



Artefacto :

- Elementos de información producidos, modificados o usados por el proceso.
- Son usados por los roles para realizar nuevas actividades y son el resultado de esas actividades.
- Ejemplo: Especificaciones funcionales, código fuente, documento de arquitectura, etc.

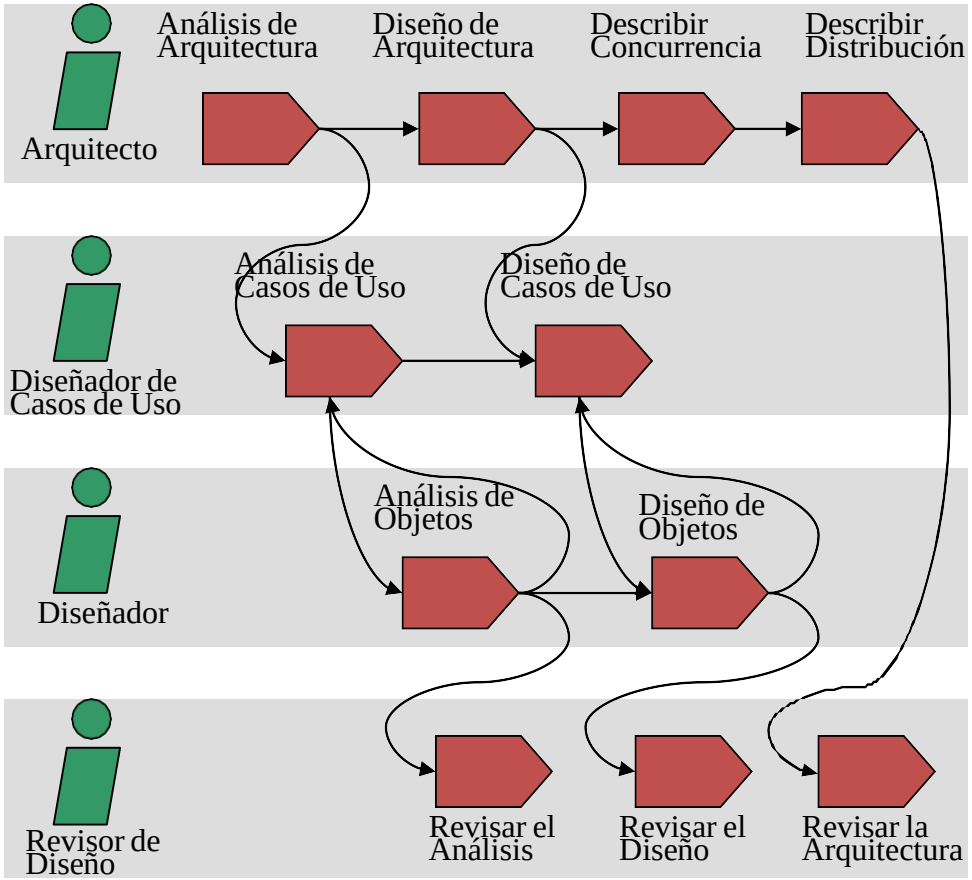
Artefactos del modelo de negocio



Flujos de trabajo:

Es una lista de actividades, roles y artefactos.

Es una secuencia de actividades que produce un resultado de valor.



¿Qué propone el RUP?

QUÉ tareas ?

- Actividades

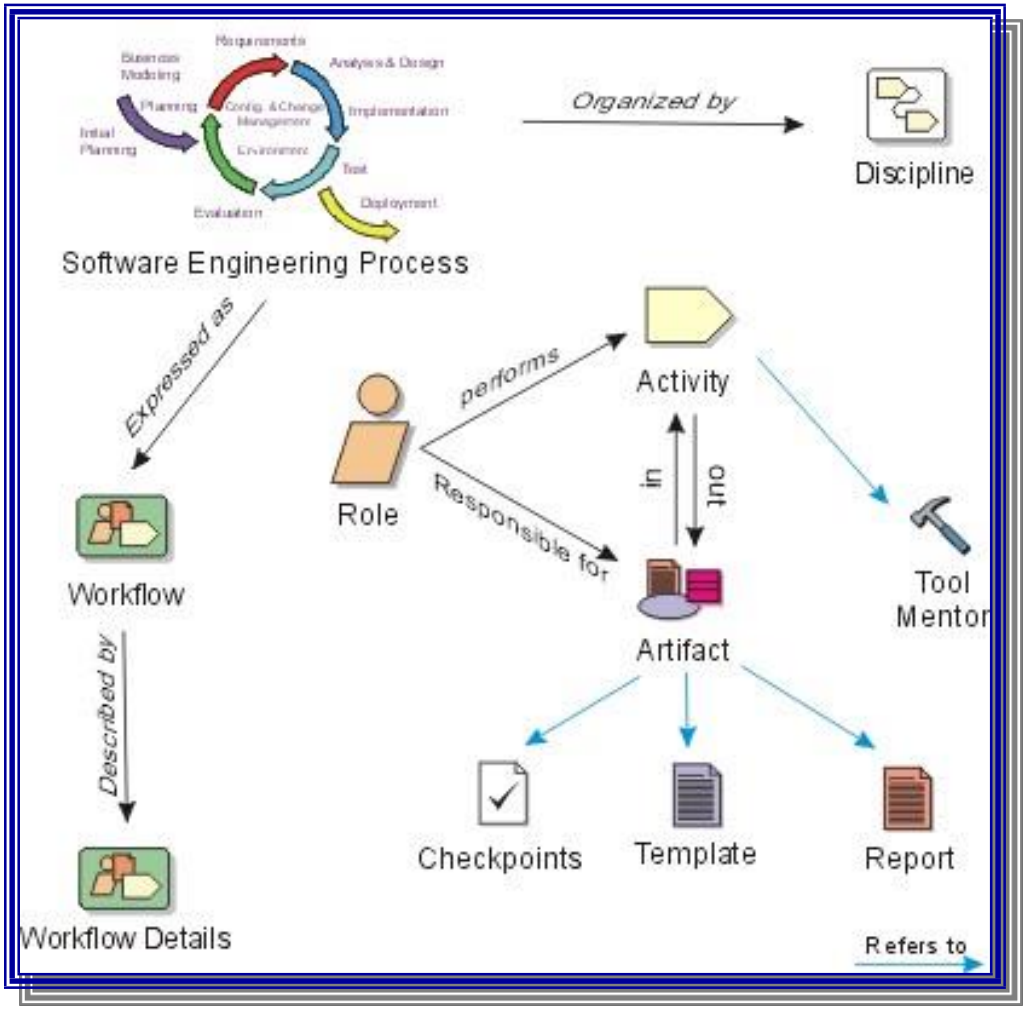
QUIÉN las hace?

- Roles

CUÁNDO se hace ?

QUÉ generar ?

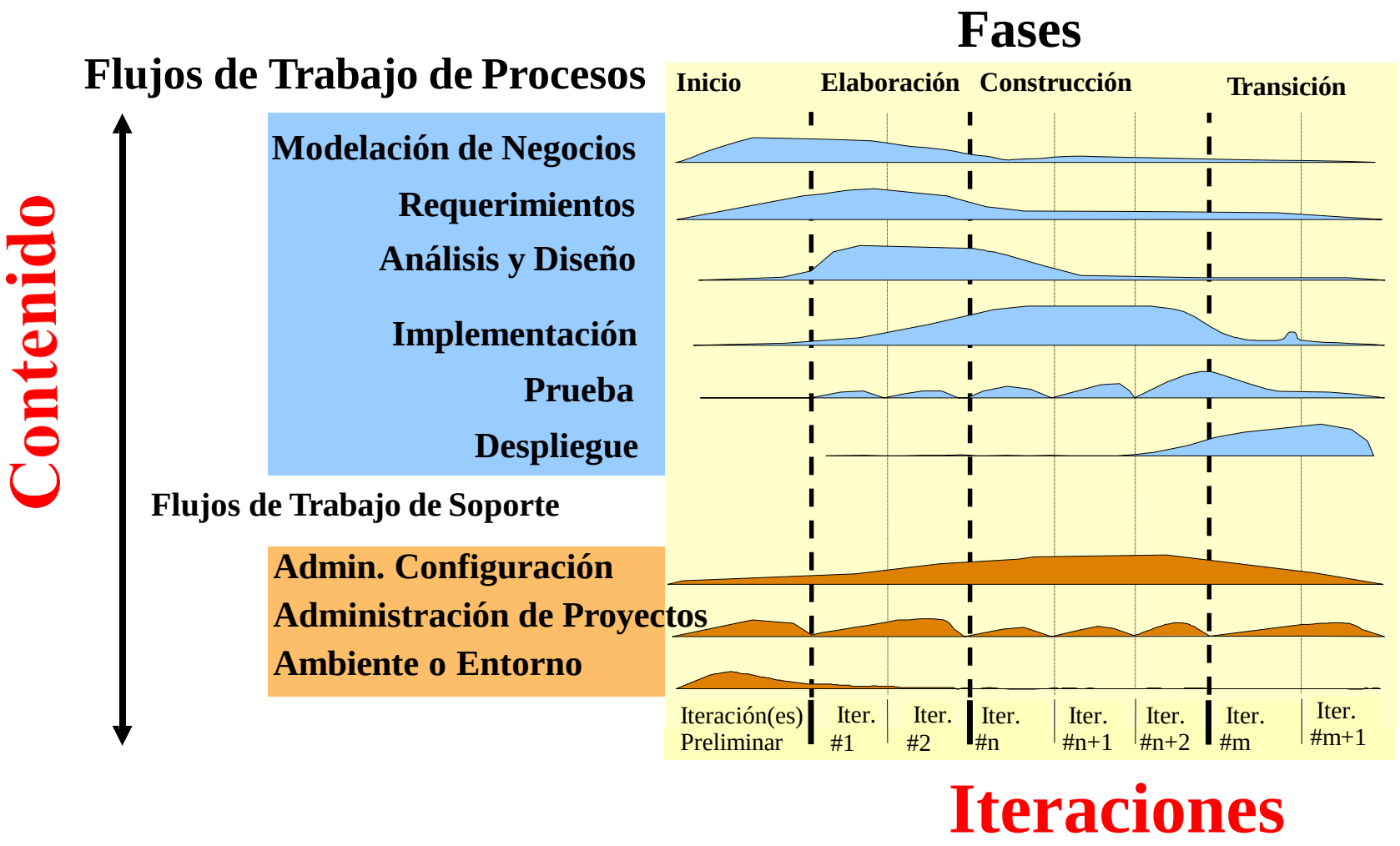
- Workflow



Estructura de RUP

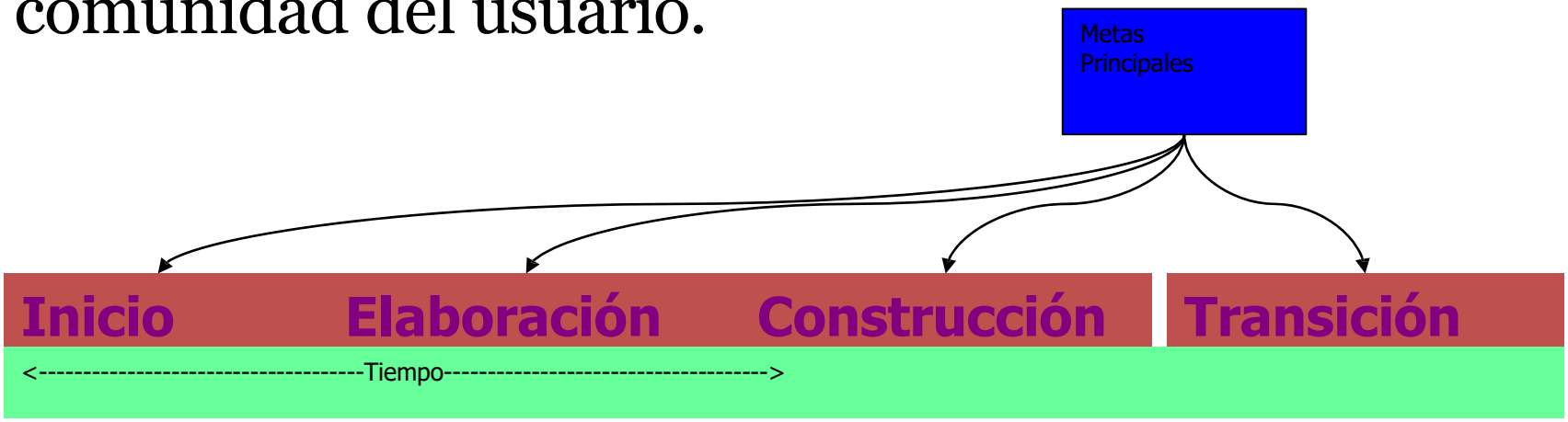
El proceso puede describirse en dos dimensiones, o a lo largo de dos ejes:

- El eje horizontal representa tiempo y muestra el aspecto dinámico del proceso, expresado en términos de ciclos, fases, iteraciones, y metas.
- El eje vertical representa el aspecto estático del proceso; como está descrito en términos de actividades, artefactos, trabajadores y flujos de trabajo.


Fases
Flujos de Trabajo de Procesos
Contenido
Modelación de Negocios
Requerimientos
Análisis y Diseño
Implementación
Prueba
Despliegue
Flujos de Trabajo de Soporte
Admin. Configuración
Administración de Proyectos
Ambiente o Entorno
Inicio
Elaboración
Construcción
Transición
Iteración(es) Preliminar
Iter. #1
Iter. #2
Iter. #n
Iter. #n+1
Iter. #n+2
Iter. #m
Iter. #m+1
Iteraciones

Fases de RUP

- **Inicio**– Define el alcance y objetivos del proyecto.
- **Elaboración** – Plan del proyecto, Especificación de características y Arquitectura base.
- **Construcción** – Construye y opera el producto.
- **Transición** – Transición del producto a la comunidad del usuario.



1 Fase de INICIO

Propósito

- Establece la propuesta técnica para un nuevo sistema o para alguna actualización importante de un sistema existente
- Especificar el alcance del proyecto
- Define el plan

2 Fase de ELABORACION

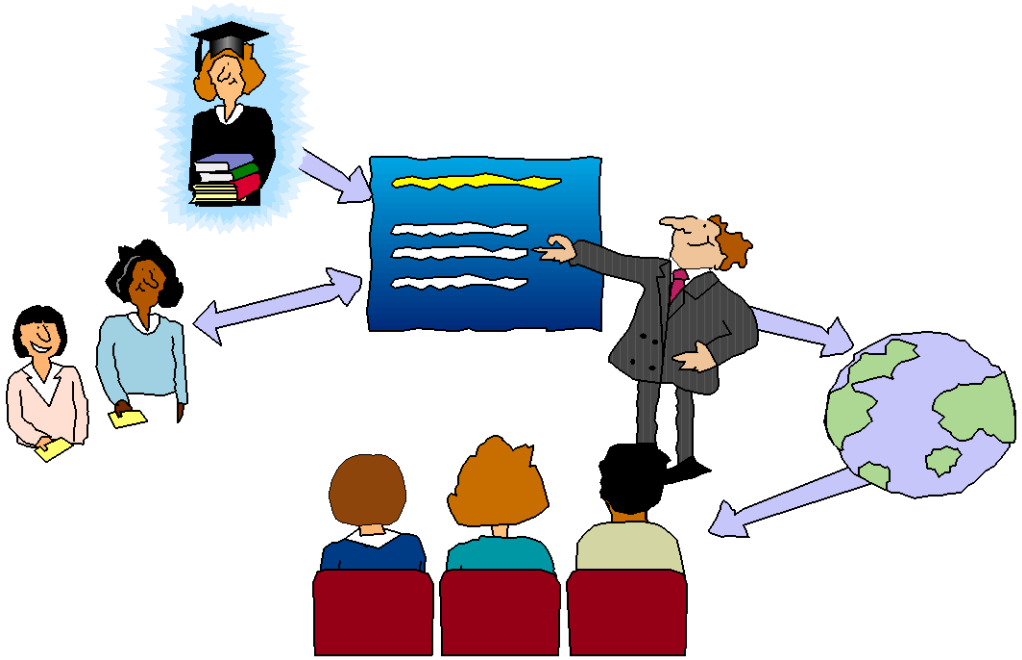
Propósito

- Analizar el dominio del problema.
- Establecer una buena arquitectura.
- Lidiar con los elementos de riesgo más altos del proyecto.
- Desarrollar un plan detallado mostrando como el proyecto será completado.

3 Fase de CONSTRUCCION

Propósito

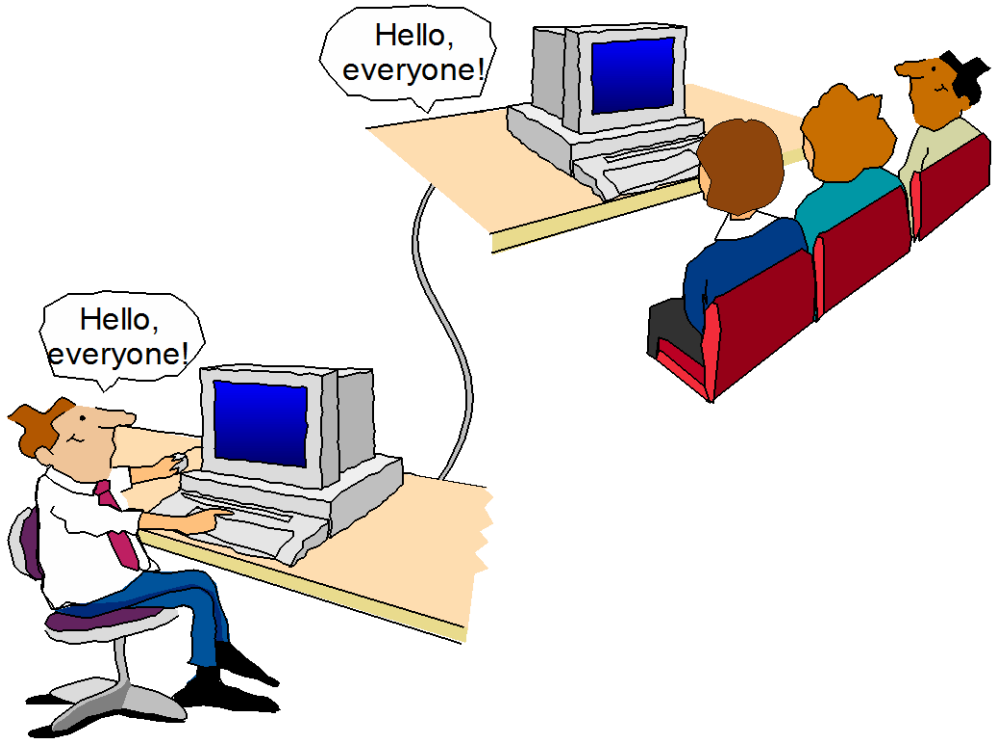
- Desarrollar incrementalmente el producto de software completo.
- Operar la aplicación



4 Fase de TRANSICION

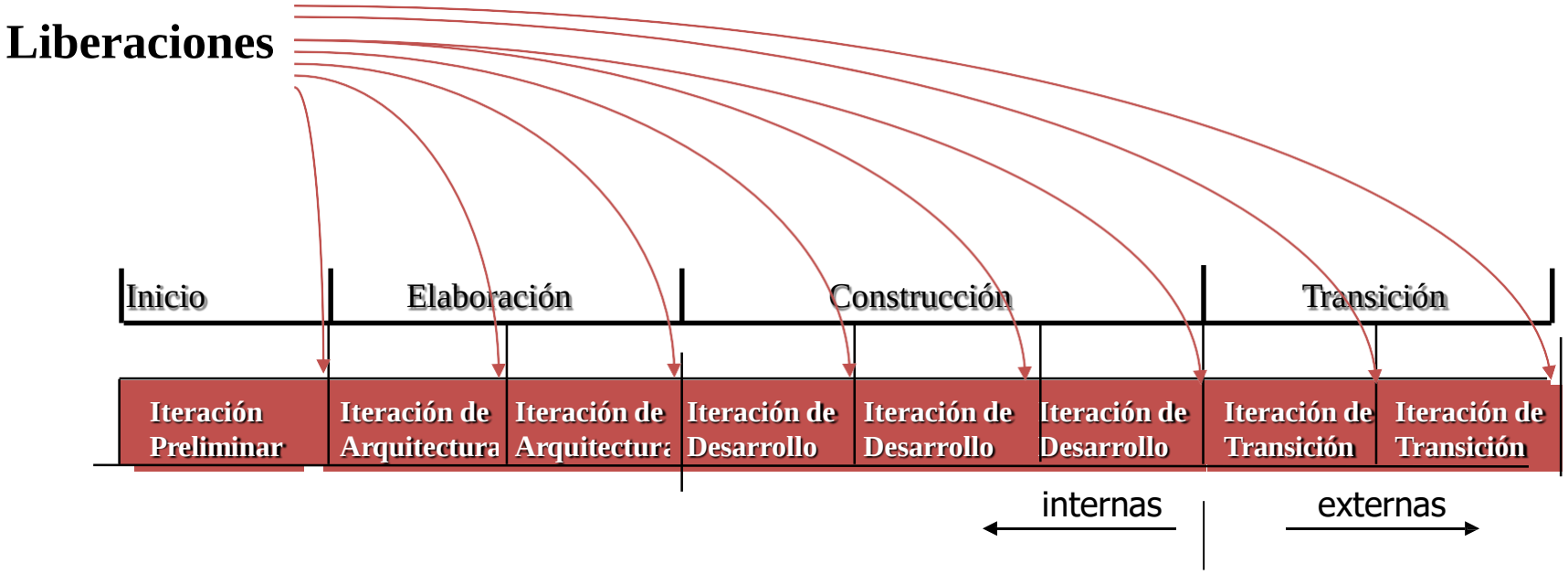
Propósito

- Hacer la transición final del producto de software al usuario.



Iteraciones

Cada fase en RUP puede descomponerse en iteraciones. Una iteración es un ciclo de desarrollo completo que genera como resultado una entrega de producto ejecutable (interna o externa).



iteraciones

Flujos de Trabajo

- Una enumeración de todos los roles, actividades y artefactos no constituyen un proceso. Se necesita una forma de describir secuencias significativas que produzcan algún resultado válido, y que muestre la interacción entre los elementos que participan.
- Un flujo de trabajo es una secuencia de actividades que producen un resultado de valor observable.

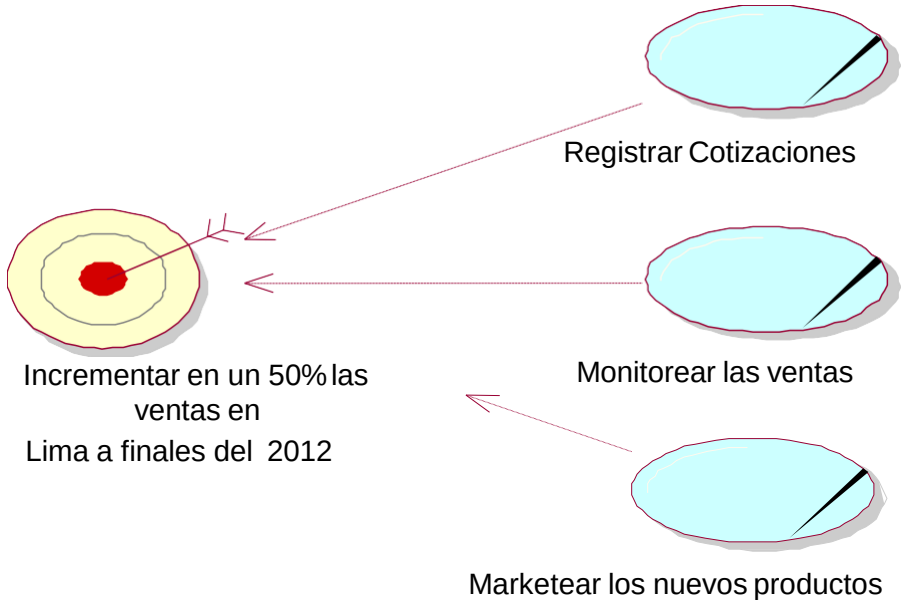
- Flujos de Trabajo para el desarrollo del sistema
  - Modelado del negocio
  - Requerimientos
  - Análisis y Diseño
  - Implementación
  - Prueba
  - Implantación

- Flujos de Trabajo para la gestión del proyecto
  - Configuración y administración de cambios
  - Administración del proyecto
  - Administración del entorno

Flujos de Trabajo para el desarrollo del sistema

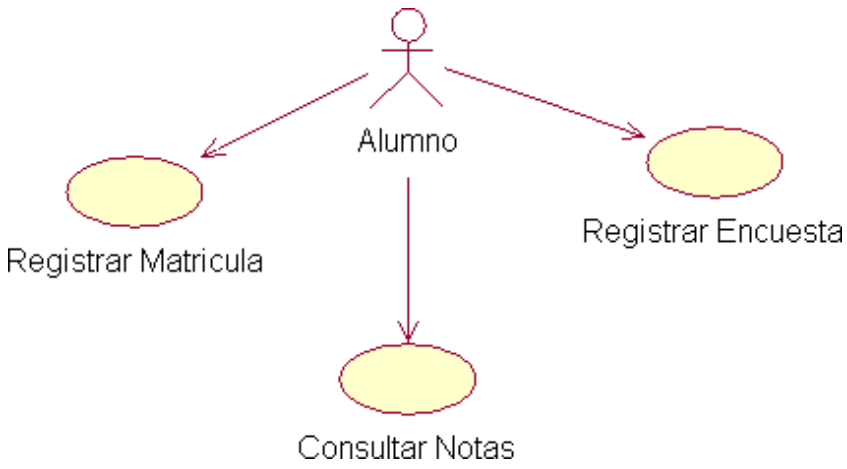
Modelo del Negocio

- Su objetivo es que el analista entienda los procesos de la empresa que son el contexto que necesita para realizar su propuesta informática.
- Asegurarse que clientes, usuarios, desarrolladores y otros involucrados tengan igual entendimiento de la empresa.



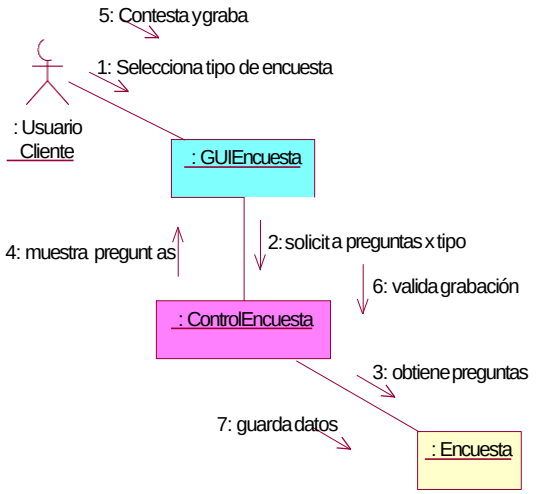
Requerimientos

- Los desarrolladores y clientes deben acordar qué es lo que el sistema debe hacer:
  - ✓Documentar funcionalidad y restricciones
  - ✓Relevar requerimientos
  - ✓Documentar decisiones
  - ✓Identificar actores
  - ✓Identificar casos de uso



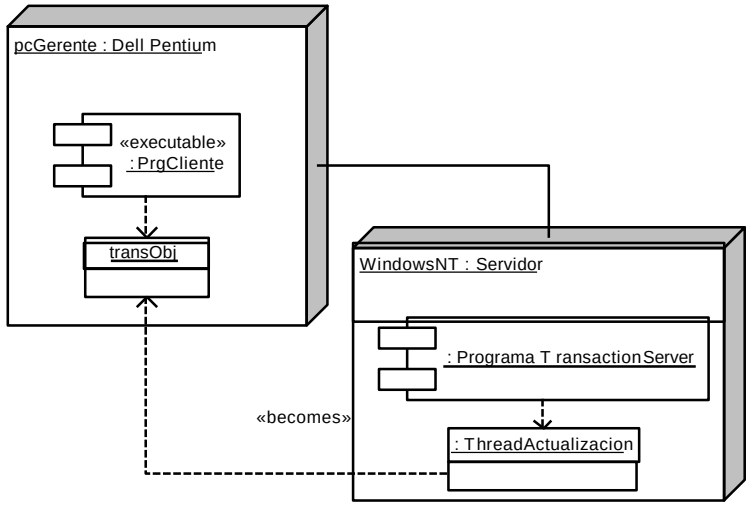
Análisis y Diseño

- Descripción de cómo se implementará el sistema: un plano
- Definición de la arquitectura tecnológica, de datos y funcional
- Identificación de los componentes
- Planteamiento de algoritmos
- Definición de patrones de diseño



Implementación

- Definir la organización del código
- Implementar clases y objetos en forma de componentes (fuente, ejecutables, etc.)
- Probar los componentes desarrollados
- Integrar los componentes en un sistema ejecutable.

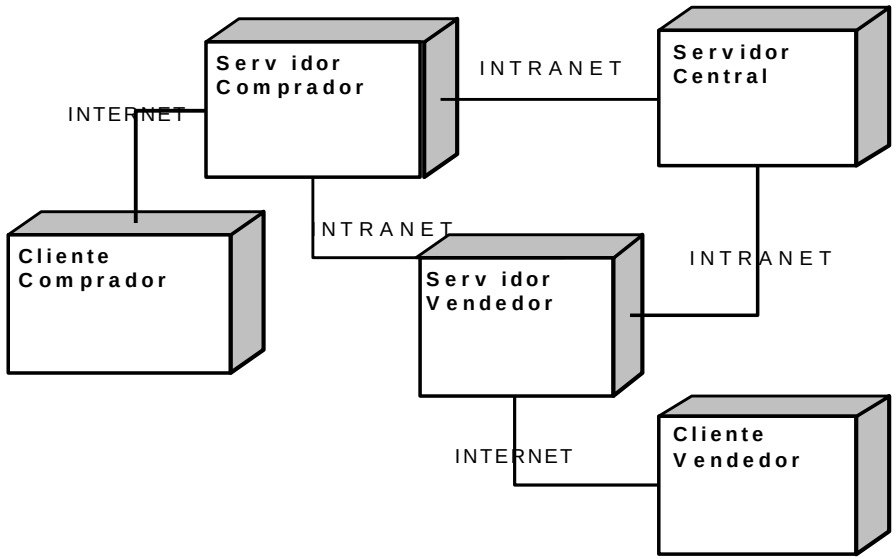


Pruebas

- Verificar la interacción entre los objetos
- Verificar la integración apropiada de componentes
- Validar que se satisfacen los requerimientos
- Identificar los defectos y corregirlos antes de la instalación.

Distribución /Despliegue

- Producir un producto y hacerlo llegar a sus usuarios finales.
- Incluye varias actividades:
  - ✓Producir un “release”
  - ✓Empaquetar el software
  - ✓Distribuir el software
  - ✓Instalar el software
  - ✓Apoyar a los usuarios



Flujos de Trabajo para la gestión del proyecto

Administración de Proyectos

- Gestiona el cumplimiento de objetivos, maneja riesgos y conduce la producción del software hacia la satisfacción de clientes y usuarios.
- Existen pocos proyectos realmente exitosos.
- RUP incluye:
  - ✓ Un framework para manejo de proyectos de software
  - ✓ Guías para planificación, provisión de personal, ejecución y monitoreo de planes
  - ✓ Un framework para manejar riesgos.

Configuración y Administración de Cambios

- Forma de controlar los artefactos producidos por las personas que trabajan en el proyecto.
- Algunos problemas habituales:
  - ✓Actualizaciones simultáneas
  - ✓Múltiples versiones
- RUP da guías para:
  - ✓Control de versiones
  - ✓Seguimiento a los cambios
  - ✓Administrar defectos

Administración del Entorno

- Prepara el ambiente y herramientas de despliegue que harán posible llevar a cabo el proyecto.
- RUP guía en la configuración de un ambiente de proceso apropiado a cada proyecto.
- Provee el soporte al equipo de desarrollo durante todo el ciclo devida del proyecto.

Repaso

- Guiarnos de las buenas prácticas de ingeniería de software, abordando las causas fundamentales.
- Las mejores prácticas se refuerzan mutuamente.
- RUP: Basado en un proceso que guía a un equipo sobre quién hace qué, cuándo y cómo.
- El Rational Unified Process es un medio para lograr mejores prácticas.

¡Gracias!