El Proceso Unificado de Desarrollo de Software

Contenidos

Objetivos de la Unidad 3

En esta unidad presentaremos el Proceso Unificado de Desarrollo de Software.

Al finalizar la unidad se espera que el alumno:

- comprenda la importancia de un proceso como marco para el desarrollo de un producto software.
- conozca y comprenda la importancia de los casos de uso como conductores del proceso de desarrollo de software.
- conozca y comprenda la importancia de la arquitectura como base para el desarrollo de un sistema software.
- conozca y comprenda la importancia de adoptar un enfoque iterativo e incremental para gestionar un proyecto de desarrollo de software.
- conozca, comprenda, y aplique los distintos flujos de trabajo del Proceso Unificado: requerimientos, análisis, diseño, codificación, y prueba.
- conozca, comprenda, y aplique como gestionar el proyecto a través de las fases de inicio, elaboración, construcción, y transición, esquema típico de las iteraciones y características de las iteraciones en cada fase del proceso.
- aplique el Proceso Unificado en el desarrollo de un sistema ejemplo experimental, aplicando herramientas orientadas a objetos y utilizando UML.

Recursos adicionales

Como principal recurso adicional que debe consultar el alumno está la bibliografía sugerida:

- EL PROCESO UNIFICADO DE DESARROLLO DE SOFTWARE
  Ivar Jacobson
  Pearson Educación

- UML y PATRONES
  Craig Larman
  Pearson Educación

1. Visión General del Proceso Unificado

Introducción

- El Proceso Unificado (RUP) es un **proceso** de desarrollo de software: "conjunto de actividades necesarias para transformar los requisitos del usuario en un sistema software".
- RUP es un **marco genérico** que puede especializarse para una variedad de tipos de sistemas, diferentes áreas de aplicación, tipos de organizaciones, niveles de aptitud y diferentes tamaños de proyectos.
- RUP está *basado en componentes*. El software esta formado por **componentes** interconectados a través de **interfaces**.
- RUP está **dirigido por casos de uso**, **centrado en la arquitectura**, y es **iterativo e incremental**.

Analizaremos un poco estas últimas tres expresiones que son fundamentales para el Proceso Unificado.

Dirigido por Casos de Uso

- Un caso de uso es un fragmento de funcionalidad del sistema que proporciona un resultado de valor a un usuario. Los casos de uso modelan los requerimientos funcionales del sistema.
- Todos los casos de uso juntos constituyen el **modelo de casos de uso**.
- Los casos de uso también **guían el proceso de desarrollo** (diseño, implementación, y prueba). Basándose en los casos de uso los desarrolladores crean una serie de modelos de diseño e implementación que llevan a cabo los casos de uso. De este modo los casos de uso no solo inician el proceso de desarrollo sino que le proporcionan un hilo conductor, avanza a través de una serie de flujos de trabajo que parten de los casos de uso.

Centrado en la Arquitectura

La **arquitectura** de un sistema software se describe mediante diferentes **vistas** del sistema en construcción.
El concepto de arquitectura software incluye los aspectos **estáticos** y **dinámicos más significativos** del sistema.
La arquitectura es una vista del diseño completo con las características más importantes resaltadas, dejando los detalles de lado.

Arquitectura: Conjunto de decisiones significativas acerca de la organización de un sistema software, la selección de los elementos estructurales a partir de los cuales se compone el sistema, las interfaces entre ellos, su comportamiento, sus colaboraciones, y su composición.

Los casos de uso y la arquitectura están profundamente relacionados. Los casos de uso deben encajar en la arquitectura, y a su vez la arquitectura debe permitir el desarrollo de todos los casos de uso requeridos, actualmente y a futuro.

El arquitecto desarrolla la forma o arquitectura a partir de la comprensión de un conjunto reducido de casos de uso fundamentales o críticos (usualmente no mas del 10 % del total). En forma resumida, podemos decir que el arquitecto:

- **ü** Crea un esquema en borrador de la arquitectura comenzando por la parte no específica de los casos de uso (por ejemplo la plataforma) pero con una comprensión general de los casos de uso fundamentales.
- **ü** A continuación, trabaja con un conjunto de casos de usos claves o fundamentales. Cada caso de uso es especificado en detalle y realizado en términos de subsistemas, clases, y componentes.
- **ü** A medida que los casos de uso se especifican y maduran, se descubre más de la arquitectura, y esto a su vez lleva a la maduración de más casos de uso.

Este proceso continúa hasta que se considere que la arquitectura es estable.

Iterativo e Incremental

Es práctico dividir el esfuerzo de desarrollo de un proyecto de software en partes más pequeñas o **mini proyectos**.
Cada mini proyecto es una **iteración** que resulta en un **incremento**.
Las iteraciones hacen referencia a pasos en el flujo de trabajo, y los incrementos a crecimientos en el producto.

Las iteraciones deben estar **controladas**. Esto significa que deben seleccionarse y ejecutarse de una forma **planificada**.
Los desarrolladores basan la selección de lo que implementarán en cada iteración en dos cosas: el conjunto de casos de uso que amplían la funcionalidad, y en los riesgos más importantes que deben mitigarse.
En cada iteración los desarrolladores identifican y especifican los casos de uso relevantes, crean un diseño utilizando la arquitectura seleccionada como guía, para implementar dichos casos de uso. Si la iteración cumple sus objetivos, se continúa con la próxima. Sino deben revisarse las decisiones previas y probar un nuevo enfoque.

Beneficios del enfoque iterativo

- La iteración controlada reduce el riesgo a los costes de un solo incremento.
- Reduce el riesgo de retrasos en el calendario atacando los riesgos más importantes primero.
- Acelera el desarrollo. Los trabajadores trabajan de manera más eficiente al obtener resultados a corto plazo.
- Tiene un enfoque más realista al reconocer que los requisitos no pueden definirse completamente al principio.

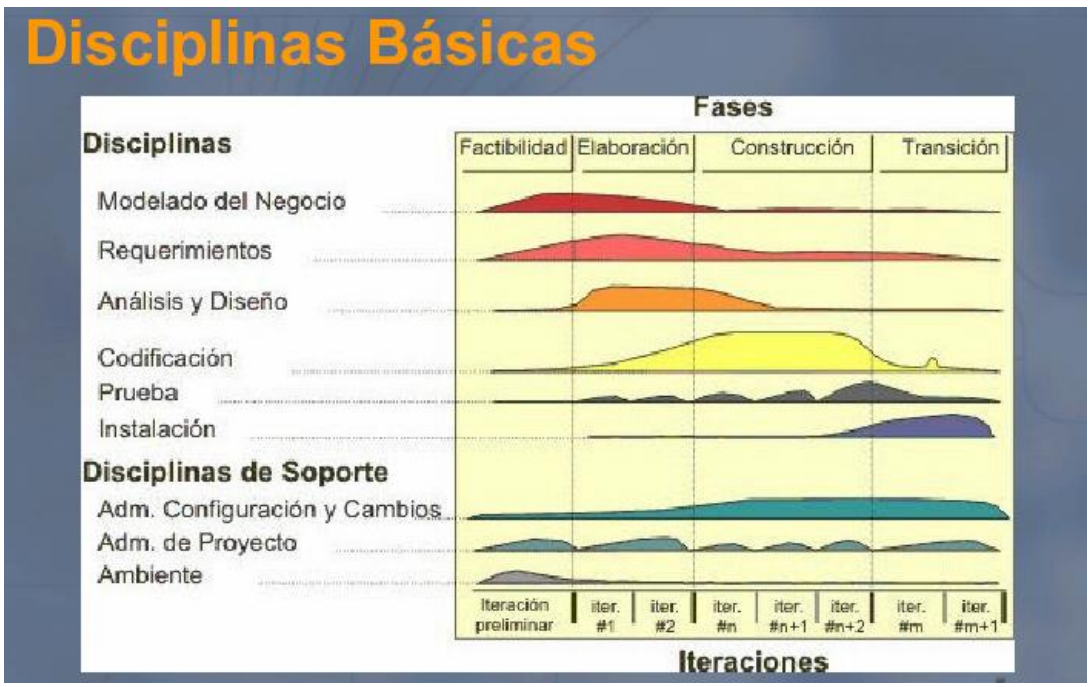
El Ciclo de Vida del Proceso Unificado

El Proceso Unificado se repite a lo largo de una serie de ciclos que constituyen la vida de un sistema. Cada ciclo constituye una **versión** del sistema.

Fases

Cada ciclo constas de cuatro fases: **inicio**, **elaboración**, **construcción**, y **transición**.

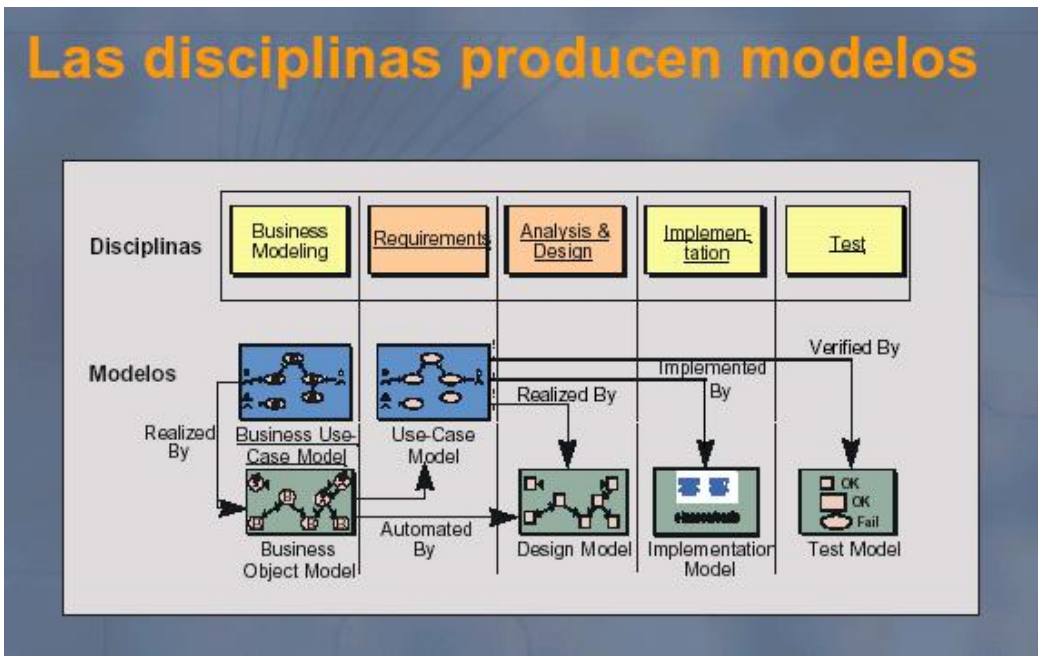
Cada fase se subdivide en **iteraciones**. En cada iteración se desarrolla en secuencia un conjunto de **disciplinas** o flujos de trabajos.



Disciplinas

Cada disciplina es un conjunto de actividades relacionadas (flujos de trabajo) vinculadas a un área específica dentro del proyecto total. Las más importantes son: **Requerimientos**, **Análisis**, **Diseño**, **Codificación**, y **Prueba**.
El agrupamiento de actividades en disciplinas es principalmente una ayuda para comprender el proyecto desde la visión tradicional en cascada.

Cada disciplina está asociada con un conjunto de **modelos** que se desarrollan. Estos modelos están compuestos por **artefactos**. Los artefactos más importantes son los modelos que cada disciplina realiza: **modelo de casos de uso**, **modelo de diseño**, **modelo de implementación**, y **modelo de prueba**.



El Proceso Unificado consiste en una serie de disciplinas o flujos de trabajo que van desde los requisitos hasta las pruebas. Los flujos de trabajo desarrollan modelos desde el modelo de casos de uso hasta el modelo de pruebas.

Disciplina	Modelos
Requisitos	Modelo de Casos de Uso
Análisis	Modelo de Análisis
Diseño	Modelo de Diseño - Modelo de Despliegue
Implementación	Modelo de Implementación
Prueba	Modelo de Prueba

Hitos

Cada fase finaliza con un **hito**. Cada hito se determina por la disponibilidad de un conjunto de artefactos, es decir un conjunto de modelos o documentos que han sido desarrollados hasta alcanzar un estado predefinido.
Los hitos tienen muchos objetivos. El más crítico es que los directores deben tomar ciertas decisiones antes de que el trabajo continúe con la siguiente fase.
Los hitos también permiten controlar la dirección y progreso del trabajo.
Al final se obtiene un conjunto de datos a partir del seguimiento del tiempo y esfuerzo consumidos en cada fase. Estos datos son útiles para realizar estimaciones en futuros proyectos.

Fase de Inicio

Durante la fase de **inicio** se desarrolla una descripción del producto final, y se presenta el **análisis del negocio**. Esta fase responde las siguientes preguntas:

- ¿Cuáles son las principales funciones del sistema para los usuarios más importantes?
- ¿Cómo podría ser la mejor arquitectura del sistema?
- ¿Cuál es el plan del proyecto y cuanto costará desarrollar el producto?

En esta fase se identifican y priorizan los **riesgos mas importantes**.

El objetivo de esta fase es ayudar al equipo de proyecto a decidir cuales son los verdaderos objetivos del proyecto. Las iteraciones exploran diferentes soluciones posibles, y diferentes arquitecturas posibles.
Puede que todo el trabajo físico realizado en esta fase sea descartado. Lo único que normalmente sobrevive a la fase de inicio es el incremento del conocimiento en el equipo.
Los artefactos que típicamente sobreviven a esta fase son:

- Un enunciado de los mayores requerimientos planteados generalmente como casos de uso.
- Un boceto inicial de la arquitectura.
- Una descripción de los objetivos del proyecto.
- Una versión muy preliminar del plan del proyecto.
- Un modelo del negocio.

La fase de inicio finaliza con el **Hito de Objetivos del Ciclo de Vida**.
Este hito es alcanzado cuando el equipo de proyectos y los stakeholders llegan a un acuerdo sobre:

- Cuál es el conjunto de necesidades del negocio, y que conjunto de funciones satisfacen estas necesidades.
- Una planificación preliminar de iteraciones.
- Una arquitectura preliminar.

Debe poder responderse las siguientes cuestiones:
- ¿Se ha determinado con claridad el ámbito del sistema? ¿Se ha determinado lo que va a estar dentro del sistema y fuera del mismo?
- ¿Se ha llegado a un acuerdo con todas las personas involucradas (stakeholders) sobre los requisitos funcionales del sistema?
- ¿Se vislumbra una arquitectura que pueda soportar estas características?
- ¿Se identifican los riesgos críticos? ¿Se prevé forma de mitigarlos?
- ¿El uso del producto justifica la relación costo-beneficio?
- ¿Es factible para su organización llevar adelante el proyecto?
- ¿Están los inversores de acuerdo con los objetivos?

Fase de Elaboración

Durante la fase de **elaboración** se especifican en detalle la mayoría de los casos de uso del producto y se diseña la arquitectura.
Las iteraciones en la fase de elaboración:
- Establecen una firme comprensión del problema a solucionar.
- Establece la fundación arquitectural para el software.
- Establece un plan detallado para las siguientes iteraciones.
- Elimina los mayores riesgos.

El resultado de esta fase es la **línea base de la arquitectura**.
En esta fase se construyen típicamente los siguientes artefactos:
- El cuerpo básico del software en la forma de un prototipo arquitectural.
- Casos de prueba
- La especificación de la mayoría de los casos de uso (80%) que describen la funcionalidad del sistema.
- Un plan detallado para las siguientes iteraciones.

La fase de elaboración finaliza con el **hito de la Arquitectura del Ciclo de Vida.**
Este hito se alcanza cuando el equipo de desarrollo y los stakeholders llegan a un acuerdo sobre:
- Los casos de uso que describen la funcionalidad del sistema.
- La línea base de la arquitectura
- Los mayores riesgos han sido mitigados
- Se ha establecido un plan del proyecto, un plan de iteraciones para las fases de construcción y transición.

Al alcanzar este hito debe poder responderse a preguntas como:
- ¿Se ha creado una línea base de la arquitectura? ¿Es adaptable y robusta? ¿Puede evolucionar?
- ¿Se han identificado y mitigado los riesgos más graves?
- ¿Se ha desarrollado un plan del proyecto hasta el nivel necesario para respaldar una agenda, costes, y calidad realistas?
- ¿Proporciona el proyecto, una adecuada recuperación de la inversión?
- ¿Se ha obtenido la aprobación de los inversores?

Fase de Construcción

Durante la fase de **construcción** se crea el producto. La línea base de la arquitectura crece hasta convertirse en el sistema completo.

Al final de esta fase, el producto contiene todos los casos de uso implementados, sin embargo puede que no este libre de defectos.
Los artefactos producidos durante esta fase son:
- El sistema software
- Los casos de prueba
- Los manuales de usuario

La fase de construcción finaliza con el **hito de Capacidad Operativa Inicial**.
Este hito se alcanza cuando el equipo de desarrollo y los stakeholders llegan a un acuerdo sobre:
- El producto es estable para ser usado
- El producto provee alguna funcionalidad de valor
- Todas las partes están listas para comenzar la transición

Fase de Transición

La fase de **transición** cubre el período durante el cual el producto se convierte en la versión beta.
Las iteraciones en esta fase continúan agregando características al software. Sin embargo las características se agregan a un sistema que el usuario se encuentra utilizando activamente.
Los artefactos construidos en esta fase son los mismos que en la fase de construcción. El equipo se encuentra ocupado fundamentalmente en corregir y extender la funcionalidad del sistema desarrollado en la fase anterior.

La fase de transición finaliza con el **hito de Lanzamiento del Producto**,.
Este hito se alcanza cuando el equipo de desarrollo y los stakeholders llegan a un acuerdo sobre:
- Se han alcanzado los objetivos fijados en la fase de Inicio.
- El usuario está satisfecho con los resultados obtenidos.

2. Un proceso conducido por Casos de Uso

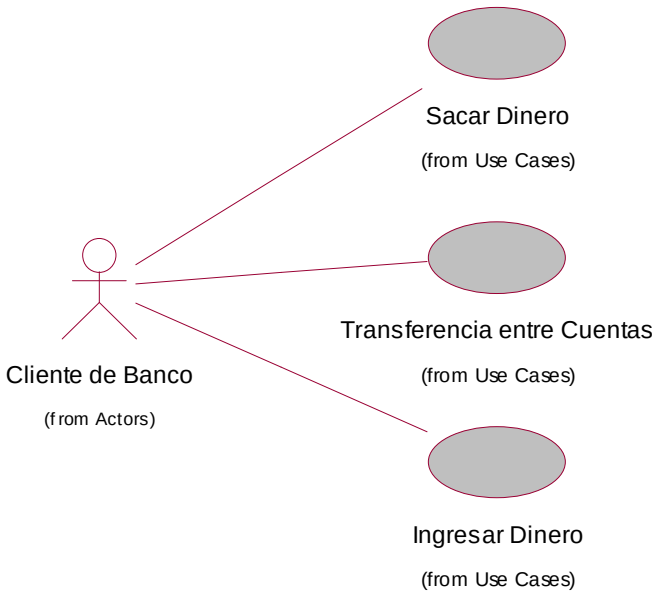
Introducción

Veamos una visión general de cómo se desarrolla el trabajo a través de todos los flujos en un proceso dirigido por casos de uso. Tomaremos como ejemplo una versión simplificada de un sistema para un Cajero Automático (CA).

1. El Modelo de Caso de Usos representa los requisitos funcionales

La primera disciplina que se desarrolla dentro de cada iteración es la de requerimientos (posiblemente luego de realizar un modelado del dominio o del negocio). El objetivo de esta fase es determinar los requerimientos del sistema. Los requerimientos funcionales son plasmados a través de casos de uso en un Modelo de Casos de Uso.
El modelo de casos de uso ayuda al cliente, a los usuarios, y a los desarrolladores a llegar a un acuerdo sobre cómo utilizar el sistema.
Cada tipo de usuario del sistema se representa mediante un **actor** que define un **rol** de utilización del sistema.
Los actores modelan el entorno del sistema, y los casos de uso especifican el sistema.
Un diagrama de casos de uso describe parte del modelo de casos de uso y muestra un conjunto de casos de uso y actores asociados.

Ej.: Modelo de Casos de Uso para el sistema Cajero Automático (CA)



Para cada caso de uso debe especificarse sus caminos o secuencias de acciones posibles.

Ej.: Secuencia de acciones para un camino del caso de uso Sacar Dinero (simplificada)

- El cliente del banco se identifica

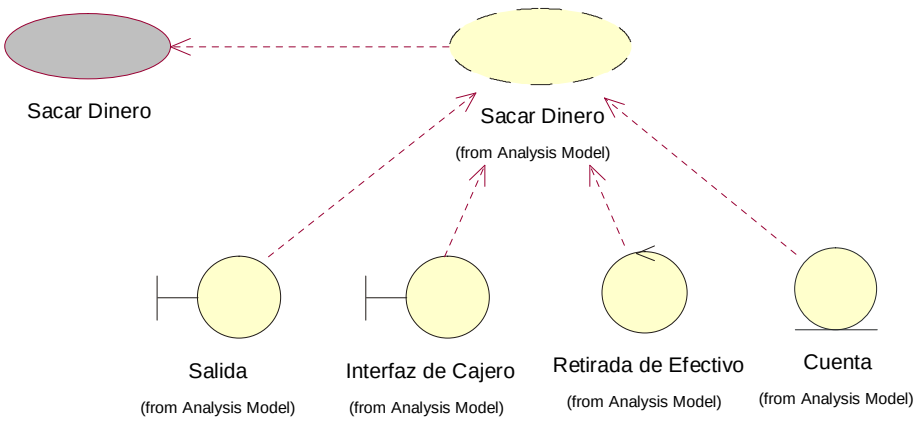
- El cliente elige de que cuenta sacar dinero y especifica cantidad
- El sistema deduce la cantidad de la cuenta y entrega el dinero

Los casos de uso también se utilizan como “contenedores” para los **requisitos no funcionales**.

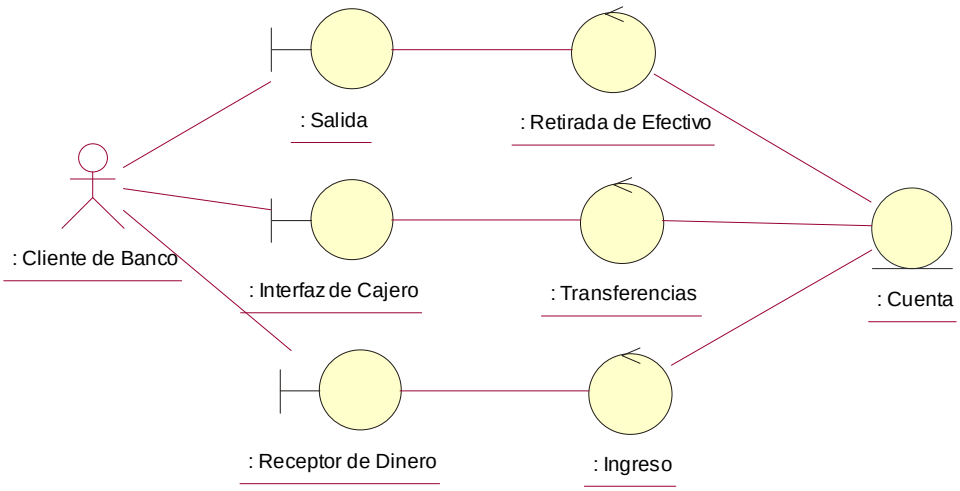
2. Creación del modelo de análisis a partir de los casos de uso

El modelo del análisis es **opcional**. En él, se describen un conjunto de Clases del Análisis que se utilizan para realizar una descripción abstracta de la **realización** de los casos de uso del modelo de casos de uso. Las clases del análisis luego evolucionan hacia otras clases más detalladas en el Modelo del Diseño.
El modelo de análisis crece incrementalmente a medida que se analizan más y más casos de uso. En cada iteración elegimos un conjunto de casos de uso y los reflejamos en el modelo de análisis. Construimos el sistema como una estructura de clasificadores (clases del análisis) y relaciones entre ellas. También describimos las colaboraciones que llevan a cabo los casos de uso, es decir las realizaciones de los casos de uso.

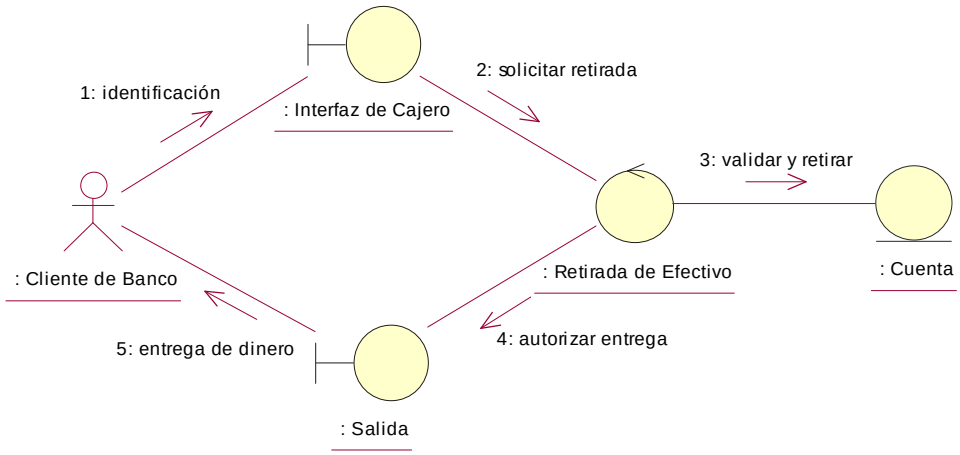
Ej.: Realización de un caso de uso en el modelo de análisis.



Ej.: Una clase puede participar en varias realizaciones.



Durante el Análisis se utilizan **diagramas de colaboración** para describir la realización de un caso de uso.



Cada clase debe cumplir todos sus roles de colaboración: las responsabilidades de una clase son sencillamente la recopilación de todos los roles que cumple en todas las realizaciones de casos de uso en las que la clase participa. Juntándolas y eliminando repeticiones entre los roles, obtenemos una especificación de todas las responsabilidades y atributos de la clase.

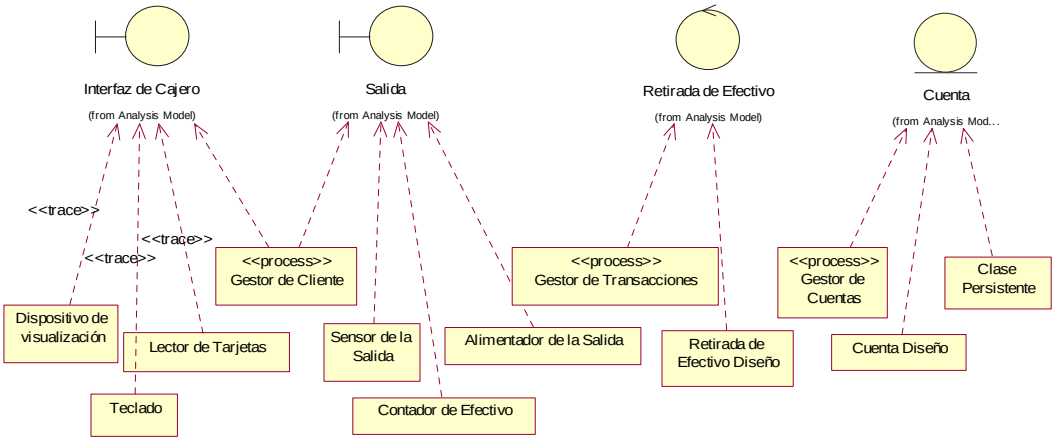
3. Creación del modelo de diseño a partir del modelo de análisis

El modelo de diseño se crea tomando el modelo de análisis como entrada principal (cuando este fue creado), y se lo adapta a un entorno de implementación particular. Esta adaptación incluye considerar por ejemplo: lenguaje a utilizar, adecuaciones a un framework de construcción de GUI particular, uso de un ORB, frameworks diversos, sistemas heredados, etc.

El modelo de diseño es similar al modelo de análisis ya que incluye *clasificadores*, *relaciones*, y *realizaciones de casos de uso*, y existe una relación de **traza** entre los artefactos del diseño y los del análisis, pero mientras estos últimos son conceptuales, los del diseño deben adecuarse al entorno de implementación específico. Por esta razón el modelo de diseño es mucho más complejo.

El modelo de diseño es una jerarquía de paquetes (subsistemas) que agrupan las clases. Los subsistemas son la “vista” del diseño de los componentes que luego se definen en el modelo de Implementación.
La jerarquía del modelo de diseño consiste de capas (layers).

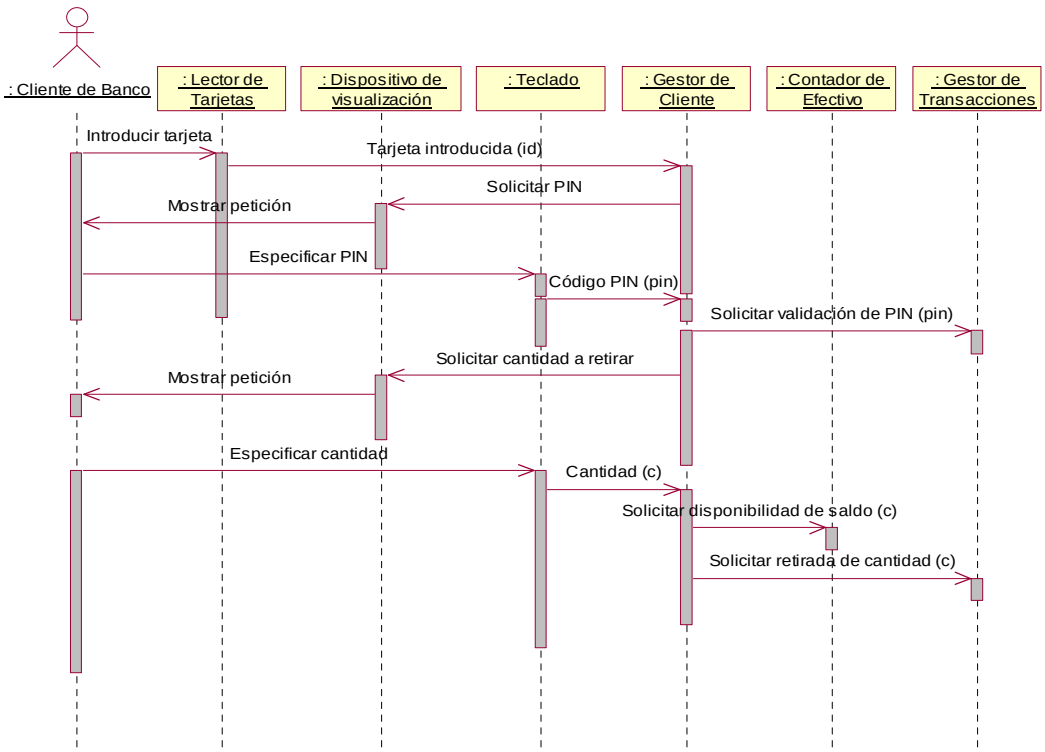
Ej.: Las clases del diseño refinan las clases del análisis.



La mayoría de las clases de diseño normalmente tienen una sola traza a una clase de análisis. Esto es habitual para las clases de diseño que son específicas de la aplicación, diseñadas para dar soporte a una aplicación o a un conjunto de ellas.

De manera similar a como lo realizamos en el análisis, debemos identificar la interacción detallada entre los objetos de diseño que tiene lugar en la realización del caso de uso en el modelo del diseño. En el diseño sin embargo, utilizaremos principalmente diagramas de secuencia para representar esta interacción.

Ej.: Diagrama de secuencia para representar la realización del caso de uso Sacar Dinero en el modelo de diseño.



Agrupación de clases en subsistemas
Es imposible utilizar sólo clases para realizar los casos de uso en un sistema grande con cientos o miles de clases. Es imposible comprender el sistema sin una organización de más alto nivel. Las clases se agrupan en subsistemas.

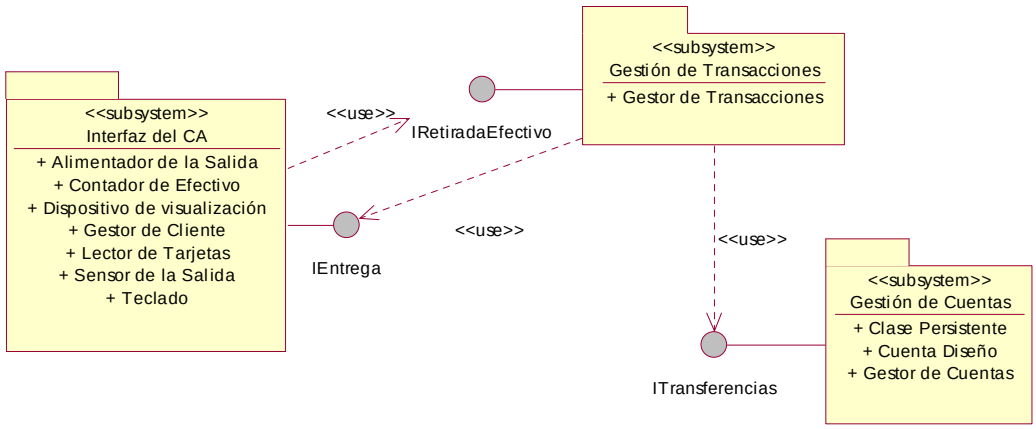
Un subsistema es un agrupamiento semánticamente útil de clases o de otros subsistemas.

Los subsistemas de bajo nivel se denominan **subsistemas de servicio**. Los subsistemas de servicio constituyen una unidad manejable de funcionalidad **opcional**.

Los subsistemas pueden diseñarse en forma **descendente** o **ascendente**.
El diseño ascendente se realiza a partir de la agrupación de clases ya identificadas. Se proponen subsistemas que empaquetan clases en unidades claramente definidas.
El diseño descendente, implica la definición previa por parte del arquitecto de los subsistemas de más alto nivel y las interfaces entre ellos, antes de que se hayan identificado las clases.

Ej.: Para el sistema de CA se agrupan las clases en tres subsistemas:
- <<subsystem>> Interfaz del CA: agrupa todas las clases que proporcionan la interfaz gráfica del CA:
  - Lector de tarjetas
  - Dispositivo de visualización
  - Teclado
  - Alimentador de la salida
  - Sensor de la salida
  - Contador de efectivo
  - **Gestor de cliente**
- <<subsystem>> Gestión de transacciones
  - **Gestión de Transacciones**
  - <<service subsystem>> Gestión de retirada de efectivo
    - Retirada de efectivo
- <<subsystem>> Gestión de cuentas
  - Clase Persistente
  - **Gestor de Cuentas**
  - Cuenta

La ventaja de colocar todas las clases de interfaz en un subsistema permitiría reemplazar el subsistema completo para adecuarlo a otra interfaz sin mayores cambios en el resto del sistema.
Los subsistemas implementan **interfaces**. Las interfaces se representan por un círculo vinculado con una línea de trazo continuo a la clase dentro del subsistema que proporciona la interfaz. Una línea de trazo discontinuo de una clase a una interfaz representa que la clase usa la interfaz.

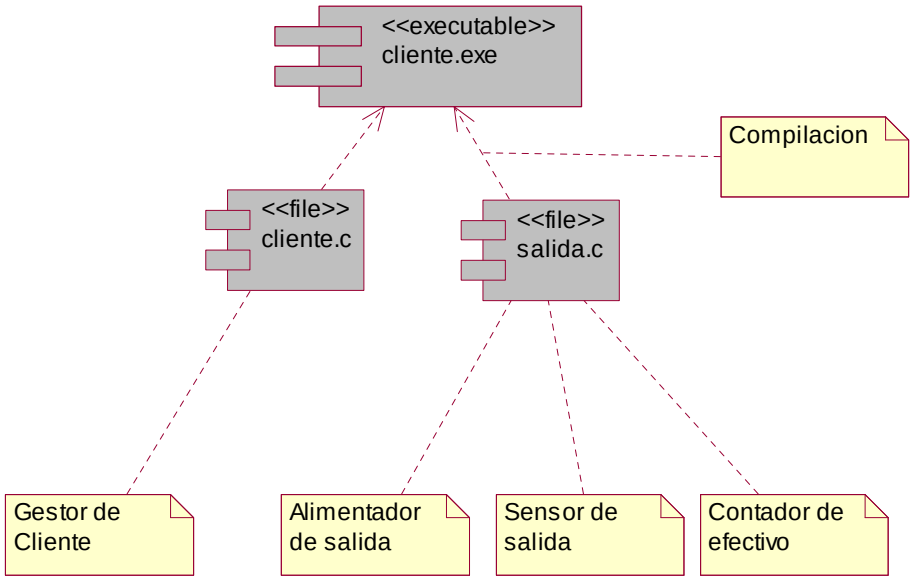


4. Creación del modelo de implementación a partir del modelo de diseño

El modelo de implementación está formado por **componentes**, que incluyen todos los **ejecutables** (Ej. ActiveX, JavaBeans, .exe), y otro tipo de componentes como ser componentes de fichero (código fuente, shell scripts, etc.), componentes de tabla (elementos de base de datos), etc.

Un **componente** es una parte física y reemplazable del sistema que cumple y proporciona la realización de un conjunto de interfaces.

Ej. Componentes implementan clases del diseño



5. Prueba de casos de uso

Durante la prueba, verificamos que el sistema implementa correctamente su especificación.
El modelo de prueba está compuesto por: casos de prueba y procedimientos de prueba.

Un **caso de prueba** es un conjunto de entradas de prueba, condiciones de ejecución, y resultados esperados, desarrollados para un objetivo concreto, tal como probar un camino concreto a través de un caso de uso, o verificar que se cumple un requisito específico.
Un **procedimiento de prueba** es una especificación de cómo llevar a cabo la preparación, ejecución, y evaluación de los resultados de un caso de prueba particular.

Ej. Un caso de prueba especifica la entrada, los resultados esperados, y otras condiciones relevantes para verificar el flujo básico del caso de uso Sacar Dinero:

Entradas:

- La cuenta 12-121-1211 del Cliente de Banco tiene un saldo de 350 $
- El Cliente de Banco se identifica correctamente
- El Cliente de Banco solicita la retirada de 200 $ de la cuenta 12-121-1211
- Hay suficiente dinero en el Cajero Automático

Resultados

- El saldo de la cuenta 12-121-1211 disminuye a 150 $
- El Cliente de Banco recibe 200 $ del Cajero Automático

Condiciones

- No se permite que ningún otro caso de uso (instancias de) acceda a la cuenta 12-121-1211 durante la ejecución del caso de prueba.

3. Un proceso centrado en la arquitectura

Introducción

Veamos una visión general de la arquitectura de un sistema de software.
La arquitectura software abarca decisiones importantes sobre:

- La organización del sistema software.
- Los elementos estructurales que compondrán el sistema y sus interfaces.
- La composición de los elementos estructurales y del comportamiento en subsistemas progresivamente más grandes
- El estilo de la arquitectura que guía esta organización: los elementos y sus interfaces, sus colaboraciones y su composición.

La arquitectura se representa mediante vistas del modelo:

- una vista del modelo de casos de uso
- una vista del modelo de análisis
- una vista del modelo de diseño
- una vista del modelo de despliegue
- una vista del modelo de implementación

Estas vistas solo tienen elementos que son **arquitectónicamente significativos**. Por Ej. La vista de los casos de uso tiene los actores y casos de uso arquitectónicamente significativos. Lo mismo sucede en los modelos de análisis y diseño.

Importancia y necesidad de una arquitectura

Se necesita una arquitectura para:

- Comprender el sistema
- Organizar el desarrollo
- Fomentar la reutilización
- Hacer evolucionar el sistema

Desarrollo de la arquitectura

La arquitectura se desarrolla mediante iteraciones, principalmente en la etapa de elaboración.
El resultado de la fase de elaboración es la **línea base de la arquitectura** – un esqueleto del sistema con pocos músculos de software.

Los casos de uso que son relevantes para la arquitectura son resumidamente aquellos que mitigan los mayores riesgos del proyecto, aquellos que son más importantes para el usuario, y aquellos que nos ayudan a cubrir todas las funcionalidades significativas.

Al final de la fase de elaboración hemos desarrollado modelos del sistema que representan los casos de uso más importantes y sus realizaciones desde el punto de vista de la arquitectura.
Esta agregación de modelos es la **línea base de la arquitectura**. Es un sistema pequeño y delgado. Tiene las versiones de todos los modelos que un sistema terminado contiene al final de la fase de construcción. Incluye el mismo esqueleto de

subsistemas, componentes y nodos que un sistema definitivo, pero no existe toda la musculatura. Es un sistema ejecutable.

Descripción de la arquitectura

La línea base de la arquitectura, es la versión interna del sistema al final de la fase de elaboración. El conjunto de modelos que describen esta línea base se denomina **Descripción de la Arquitectura**.

El papel de la descripción de la arquitectura es guiar al equipo de desarrollo a través del ciclo de vida del sistema.
La descripción de la arquitectura puede adoptar diferentes formas. Puede ser un extracto de los modelos que son parte de la línea base de la arquitectura, o puede ser una reescritura de los extractos de forma que sea más fácil leerlos.

La descripción de la arquitectura tiene cinco secciones, una para cada modelo: una vista del modelo de casos de uso, una vista del modelo de análisis (opcional / descartable), una vista del modelo de diseño, una vista del modelo de despliegue, y una vista del modelo de implementación.

La vista de la arquitectura del modelo de casos de uso
Presenta los actores y casos de uso más importantes.

Ej. Vista de la arquitectura del modelo de casos de uso del sistema CA
En el ejemplo del CA el caso de uso más importante es Sacar Dinero. Sin él, no tendría sentido el CA.
Para definir la arquitectura por tanto, el arquitecto sugiere que el caso de uso Sacar Dinero se implemente en su totalidad durante la fase de elaboración.

La vista de la arquitectura del modelo de diseño
Presenta los clasificadores más importantes para la arquitectura pertenecientes al modelo de diseño: los subsistemas e interfaces más importantes, así como algunas pocas clases muy importantes, fundamentalmente clases activas.
También presentan como se realizan los casos de uso en términos de esos clasificadores.

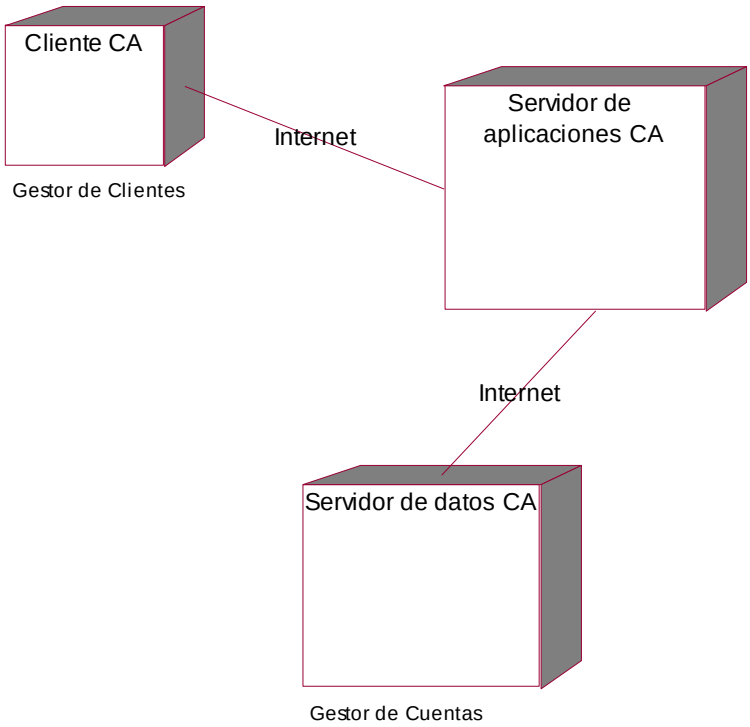
Ej. Vista de la arquitectura del modelo de diseño CA
Se incluyen las tres clases activas: Gestor de Clientes, Gestor de Transacciones, y Gestor de Cuentas.
También se incluyen los subsistemas: Interfaz del CA, Gestión de Transacciones, y Gestión de Cuentas, por ser necesarios para la realización del caso de uso Sacar Dinero.

La vista de la arquitectura del modelo de despliegue
Presenta los nodos interconectados y las clases activas que se ejecutan en ellos identificados durante el diseño.
Esto puede mostrarse por diagramas de despliegue.

Ej. Vista de la arquitectura del modelo de despliegue CA
Se incluyen los siguientes nodos y objetos activos:

- Nodo :Cliente CA – Objeto activo :Gestor de Clientes
- Nodo :Servidor de aplicaciones CA – Objeto activo :Gestor de transacciones

- Nodo :Servidor de datos CA – Objeto activo :Gestor de Cuentas



La vista de la arquitectura del modelo de implementación
El modelo de implementación es una correspondencia directa de los modelos de diseño y de despliegue.
Cada subsistema de servicio del diseño normalmente termina siendo un componente por cada tipo de nodo en el que deba instalarse.

4. Un proceso iterativo e incremental

Desarrollo en pequeños pasos

La tercera clave importante del RUP consiste en desarrollar un producto software en pasos pequeños manejables:

- Planificar un poco.
- Especificar, diseñar, e implementar un poco.
- Integrar, probar, y ejecutar un poco en cada iteración.

Las iteraciones en las primeras fases tratan en su mayor parte con la determinación del ámbito del proyecto, la eliminación de los riesgos críticos, y la creación de la línea base de la arquitectura. Después, a medida que avanzamos a lo largo del proyecto y vamos reduciendo gradualmente los riesgos restantes e implementado los componentes, la forma de las iteraciones cambia, dando incrementos como resultados.

¿Por qué un desarrollo iterativo e incremental?

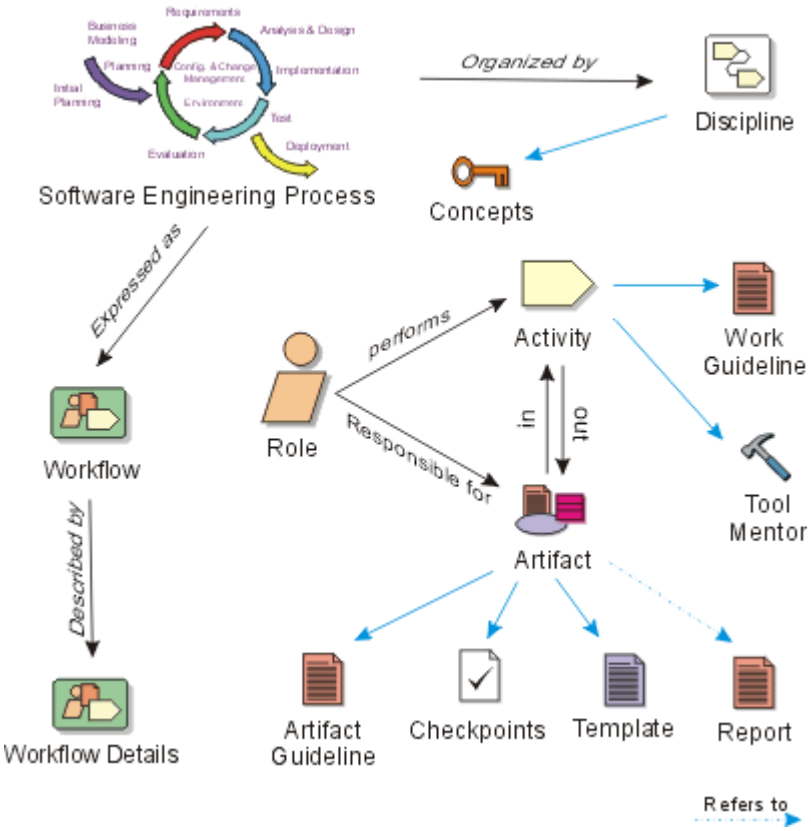
- Para tomar las riendas de los riesgos críticos y significativos desde el principio.
- Para poner en marcha una arquitectura que guíe el desarrollo del software.
- Para proporcionar un marco de trabajo que gestione de mejor forma los inevitables cambios en los requisitos y en otros aspectos.
- Para construir el sistema a lo largo del tiempo en lugar de hacerlo de una sola vez cerca del final, cuando el cambiar algo se ha vuelto costoso.
- Para proporcionar un proceso de desarrollo a través del cual el personal puede trabajar de manera más eficaz.

La iteración genérica

Una iteración es un mini proyecto, un recorrido más o menos completo a lo largo de todos los flujos de trabajo fundamentales, que obtiene como resultado una versión interna del sistema.

5. Conceptos clave

Conceptos Claves



Proceso de Ingeniería de Software

Un proceso es un conjunto de pasos ordenados para alcanzar un objetivo. En ingeniería de software, el objetivo es construir un producto de software nuevo o extender uno existente. En RUP esto se organiza en un conjunto de Disciplinas que define un flujo de trabajo.

Disciplina

Una disciplina es una colección de actividades relacionadas vinculadas con un área específica del proyecto. Este agrupamiento de actividades en disciplinas es principalmente para facilitar la comprensión del proyecto desde la perspectiva tradicional del modelo en cascada.

Flujo de trabajo

Un flujo de trabajo describe la secuencia en que se realizan las actividades en una disciplina, quienes la realizan (trabajadores) y que artefactos producen.

Trabajador (Rol)

Un trabajador o **rol**, define un comportamiento o responsabilidades de un individuo o grupo de individuos trabajando en equipo, en el contexto de una organización de ingeniería de software.

Actividad

Los trabajadores realizan actividades. Una actividad es algo que realiza un trabajador para proveer un resultado de valor en el contexto de un proyecto.

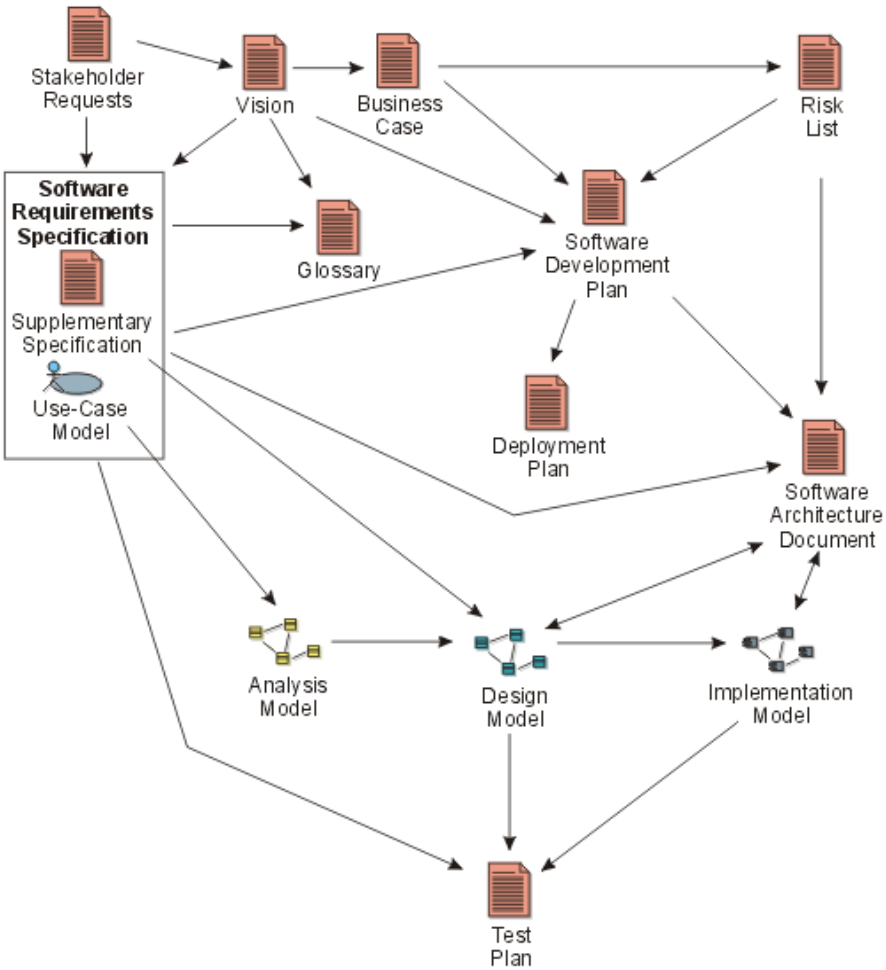
Pasos

Las actividades son descompuestas en pasos. Podemos distinguir tres categorías de pasos:

- Pasos de análisis: donde el trabajador comprende la naturaleza de la tarea, examina los artefactos de entrada, y formula las salidas.
- Pasos de acción: donde los trabajadores crean o actualizan algunos artefactos.
- Pasos de revisión: donde los trabajadores inspeccionan los resultados según determinados criterios.

Artefactos

Las actividades tienen artefactos de entrada y de salida. Un artefacto es un producto de trabajo en un proceso: los trabajadores utilizan artefactos para realizar actividades y producen artefactos como resultado de sus actividades. Los artefactos son responsabilidad de un único trabajador y promueven la idea de que toda pieza de información en el proceso debe ser responsabilidad de un rol específico. Un trabajador es el "propietario" de un artefacto, pero otros trabajadores pueden usarlo y tal vez modificarlo si tienen permiso para ello.



Principales artefactos en el proceso y flujo de información entre ellos.
El diagrama muestra como la información fluye a través del proyecto vía los artefactos.
Por claridad se han omitido muchos artefactos

Resumen de Trabajadores, Actividades, y Artefactos en las Disciplinas

Actividad	Trabajador	Responsable de artefacto (Salida)	Artefactos (entrada)
REQUISITOS			
1 Encontrar actores y casos de uso	Analista de Sistemas	Modelo de casos de uso (esbozado)	Modelo del negocio o Modelo del dominio
		Glosario	Requisitos adicionales
			Lista de características
2 Priorizar casos de uso	Arquitecto	Descripción de la arquitectura (vista del mod.de casos de uso)	Modelo de casos de uso (esbozado)
			Requisitos adicionales
			Glosario
3 Detallar Casos de Uso	Especificador de casos de uso	Caso de uso (detallado)	Modelo de casos de uso (esbozado)
			Requisitos adicionales
			Glosario
4 Estructurar el modelo de casos de uso	Analista de Sistemas	Modelo de casos de uso (estructurado)	Modelo de casos de uso (esbozado)
			Requisitos adicionales
			Caso de uso (detallado)
			Glosario
5 Prototipar interfaz de usuario	Diseñador de Interfaz de usuario	Prototipo de interfaz de usuario	Modelo de casos de uso
			Requisitos adicionales
			Caso de uso (detallado)
			Glosario
ANALISIS			
1 Análisis de la arquitectura	Arquitecto	Paquete del análisis (esbozado)	Modelo de casos de uso
		Clase del análisis (esbozada)	Requisitos adicionales
		Descripción de la arquitectura (vista del modelo de análisis)	Modelo del negocio o modelo del dominio
			Descripción de la arquitectura (vista del modelo de casos de uso)
2 Analizar un caso de uso	Ingeniero de casos de uso	Realización de caso de uso análisis	Modelo de casos de uso
		Clase del análisis (esbozo)	Requisitos adicionales
			Modelo del negocio o modelo del dominio
			Descripción de la arquitectura (vista del modelo de análisis)

3 Analizar una clase	Ingeniero de componentes	Clase del análisis (terminada)	Realización de caso de uso-análisis
			Clase del análisis (esbozo)
4 Analizar un paquete	Ingeniero de componentes	Paquete del análisis (terminado)	Descripción de la arquitectura (vista del modelo de análisis)
			Paquete del análisis (esbozo)
D I S E Ñ O			
1 Diseño de la arquitectura	Arquitecto	Subsistema (esbozado)	Modelo de casos de uso
		Interfaz (esbozada)	Requisitos adicionales
		Clase de diseño (esbozada)	Modelo de análisis
		Modelo de despliegue (esbozado)	Desc.de la arq. (vista del modelo de análisis)
		Desc.de la arq. (vista de los modelos de diseño y distr.)	
2 Diseño de un caso de uso	Ingeniero de casos de uso	Realización de caso de uso (diseño)	Modelo de casos de uso
		Clase de diseño (esbozada)	Requisitos adicionales
		Subsistema (esbozado)	Modelo de análisis
		Interfaz (esbozada)	Modelo de diseño
			Modelo de despliegue.
3 Diseño de una clase	Ingeniero de componentes	Clase de diseño (terminada)	Realización de casos de uso – diseño
			Clase de diseño (esbozada)
			Interfaz (esbozada)
			Clase del análisis (terminada)
4.Diseño de un subsistema	Ingeniero de componentes	Subsistema (terminado)	Desc. de la arquitectura (vista del mod. de diseño)
		Interfaz (terminada)	Subsistema (esbozado)
			Interfaz (esbozada)
I M P L E M E N T A C I O N			
1 Implementación de la arquitectura	Arquitecto	Componente (esbozado y posib.asignado a nodos)	Modelo de diseño
		Descripción de la arquitectura (vista de los modelos de impl.y despl.)	Modelo de despliegue
			Descrip.de la arquitectura (vista de los mod.de diseño y despliegue)
2 Integrar el sistema	Integrador de sistemas	Plan de integración de construcciones	Requisitos adicionales
		Modelo de implementación (construcciones anteriores)	Modelo de casos de uso
			Modelo de diseño
			Modelo de implementación (construcciones anteriores)
3 Implementar un subsistema	Ingeniero de componentes	Subsistema de implementac. (implementado para una construcción)	Plan de integración de construcciones
		Interfaz (implementado para una construcción)	Descrip.de la arquitec. (vista del modelo de implementac)
			Subsistema de diseño (terminado)

			Interfaz (terminado)
4 Implementar una clase	Ingeniero de componentes	Componente (implentado)	Clase de diseño (terminada)
			Interfaz (proporcionada por la clase de diseño)
5 Realizar prueba de unidad	Ingeniero de componentes	Componente (probado)	Componente (implementado)
			Interfaz
P R U E B A			
1 Planificar prueba	Ingeniero de pruebas	Plan de Prueba	Requisitos adicionales
			Modelo de casos de uso
			Modelo de análisis
			Modelo de diseño
			Modelo de implementación
			Descripción de arqu.(vistas arquitec.de los modelos)
2 Diseñar prueba	Ingeniero de pruebas	Caso de prueba	Requisitos adicionales
		Procedimiento de prueba	Modelo de casos de uso
			Modelo de análisis
			Modelo de diseño
			Modelo de implementación
			Descripción de arqu.(vistas arquitec.de los modelos)
			Plan de prueba (estrategia de prueba y planific.)
3 Implementar prueba	Ingeniero de componentes	Componente de prueba	Caso de prueba
			Procedimiento de prueba
			Modelo de implement. (construida para ser probada)
4 Realizar pruebas de integración	Ingeniero de pruebas de integración	Defecto	Caso de prueba
			Procedimiento de prueba
			Componente de prueba
			Modelo de implementación (construcción a probar)
5 Realizar prueba de sistema	Ingeniero de pruebas de sistema	Defecto	Caso de prueba
			Procedimiento de prueba
			Componente de prueba
			Modelo de implementación (construcción a probar)
6 Evaluar prueba	Ingeniero de pruebas	Evaluación de prueba (para una iteración)	Plan de prueba
			Modelo de prueba
			Defecto

6. Captura de Requisitos

Introducción

El propósito fundamental del flujo de trabajo de los requisitos es guiar el desarrollo hacia el sistema correcto.
Hay diferentes puntos de partida para la captura de requisitos. En algunos casos comenzamos haciendo un ***modelo del negocio*** o partimos de uno ya desarrollado. En otros casos si es un sistema acotado que no da soporte al negocio podemos partir de un modelo de objetos sencillo como un ***modelo del dominio***.
En otros casos el cliente puede ya haber desarrollado una especificación completa de requisitos no basada en objetos, de la cual podemos partir.

En forma típica, el flujo de trabajo de requisitos incluye los siguientes pasos:

- Enumerar los requisitos candidatos.
- Comprender el contexto del sistema.
- Capturar requisitos funcionales.
- Capturar requisitos no funcionales.

Enumerar los requisitos candidatos

La lista de características deseadas del sistema constituyen los requisitos candidatos.
De cada característica se registra:

- Nombre corto
- Descripción
- Estado (propuesto, aprobado, incluido, o validado)
- Coste estimado de implementación (en término de tipos de recursos y horas-hombre)
- Prioridad (crítico, importante, o secundario)
- Nivel de riesgo asociado a la implementación de la característica (crítico, significativo, ordinario)

Estos valores se utilizan para estimar el tamaño del proyecto y decidir cómo dividirlo en secuencia de iteraciones. La prioridad y nivel de riesgo asociados por ejemplo, se utiliza para decidir en que iteración se implementará la característica.

Comprender el contexto del sistema

Hay por lo menos dos aproximaciones para expresar el contexto de un sistema: modelado del dominio y modelado del negocio.
Un modelo del dominio describe los conceptos importantes del contexto como objetos del dominio relacionados entres sí.
Un modelo del negocio es más amplio. Describe los procesos con el objetivo de comprenderlos. El modelado del negocio especifica que **procesos de negocio** soportará el sistema.

Capturar requisitos funcionales

Los requisitos funcionales son capturados por medio de casos de uso, que conforman el modelo de casos de uso. Los casos de uso también capturan requisitos no funcionales específicos de un caso de uso determinado.

Capturar requisitos no funcionales
Los requisitos no funcionales especifican propiedades del sistema, como restricciones del entorno o de la implementación, rendimientos, etc.
Hay requisitos no funcionales específicos para un caso de uso y otros genéricos para la aplicación. Los que son específicos para un caso de uso, pueden documentarse junto con el caso de uso correspondiente. Los que son más genéricos se documentan por medio de una **lista de requisitos adicionales**.

Modelo del Dominio

Un modelo del dominio captura los tipos más importantes de objetos en el contexto del sistema. Los objetos del dominio representan las "cosas" que existen o los eventos que suceden en el entorno en el que trabaja el sistema.
Las clases del dominio aparecen en tres formas típicas:

- Objetos del negocio que representan cosas que se manipulan en el negocio, como pedidos, cuentas, contratos, etc.
- Objetos del mundo real y conceptos de los que el sistema debe hacer seguimiento como aviación enemiga, misiles, trayectorias, etc.
- Sucesos que ocurrirán o han ocurrido, como llegada de un avión, su salida, hora de la comida, etc.

El modelo de dominio se representa fundamentalmente por diagramas de clases en UML.

El objetivo del modelado del dominio es comprender y describir las clases más importantes dentro del contexto del sistema.

Modelo del Negocio

El modelado del negocio es una técnica para comprender los procesos de negocio de la organización.
El modelado del negocio está soportado por dos tipos de modelos de UML: el modelado de casos de uso, y modelos de objetos.

Un Modelo de Casos de Uso del Negocio describe los procesos de negocio de una empresa en términos de *casos de uso del negocio* y *actores del negocio* que se corresponden con los *procesos* del negocio y los *clientes* respectivamente.
Al igual que el modelo de casos de uso para un sistema software, el modelo de casos de uso del negocio presenta un sistema (en este caso, el negocio) desde la perspectiva de su uso, y esquematiza como proporciona valor a sus usuarios.
El modelo de casos de uso del negocio se describe mediante diagramas de casos de uso.

Un modelo de objetos del negocio describe como cada caso de uso del negocio es llevado a cabo por parte de un conjunto de *trabajadores* que utilizan un conjunto de *entidades del negocio* y de *unidades de trabajo*.
Cada realización de un caso de uso del negocio puede mostrarse en diagramas de interacción y diagramas de actividad.

Una **entidad** del negocio representa algo que los trabajadores toman, manipulan, inspeccionan, producen o utilizan en un negocio.
Una **unidad de trabajo** es un conjunto de esas entidades que conforma un todo reconocible para el usuario final.

La técnica de modelado de negocio identifica **entidades** y **trabajadores** que participan en la realización de los casos de uso del negocio.
Los trabajadores identificados en el modelo de negocio se utilizan como punto de partida para derivar un primer conjunto de actores y casos de uso del sistema.

Búsqueda de Casos de Uso a partir de un modelo del negocio

En primer lugar se identifica un actor por cada trabajador y por cada actor del negocio (es decir, el cliente).
Cada trabajador y actor del negocio que vaya a ser usuario del sistema de información requerirá un soporte por parte del mismo. El soporte necesario se determina tratando cada uno de los actores uno detrás de otro.

Una vez que hemos encontrado todos los roles de un trabajador o actor del negocio, uno por cada caso de uso del negocio en el que participa, podemos encontrar los casos de uso de los actores del sistema.

La manera más directa de identificar un conjunto tentativo de casos de uso es crear un caso de uso para el actor correspondiente a cada rol de cada trabajador y de cada actor del negocio. Los analistas pueden después ajustar los casos de uso tentativos.

Requisitos adicionales

Los requisitos adicionales, son requerimientos **no funcionales** que no pueden asociarse a ningún caso de uso en particular. Algunos ejemplos son el rendimiento, las interfaces, y los requisitos de diseño físico, así como las restricciones arquitectónicas.
Los requisitos adicionales se capturan de forma muy parecida a como se hacía en la especificación de requisitos tradicional, es decir con una lista de requisitos. Luego se utilizan durante el análisis junto al modelo de casos de uso.

Captura de requisitos como casos de uso

Los requisitos funcionales se estructuran de forma natural mediante casos de uso que constituyen el Modelo de Casos de Uso.
Los requisitos no funcionales restantes, se modelan como dijimos en el documento de requisitos adicionales.

Trabajadores y Artefactos

Trabajador	Responsable de (artefacto)
Analista de sistemas	Modelo de casos de uso
	Actores
	Glosario
Especificador de casos de uso	Caso de uso
Diseñador de Interfaz de Usuario	Prototipo de interfaz de usuario
Arquitecto	Descripción de la arquitectura (vista del modelo de casos de uso)

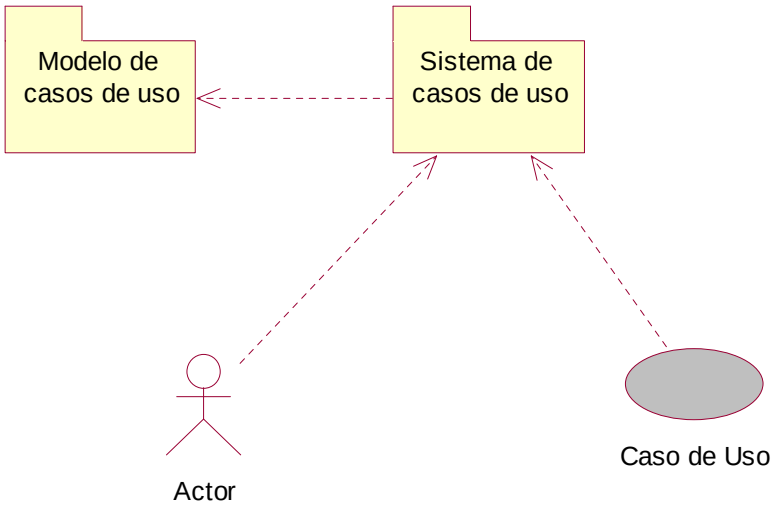
Artefactos

Los artefactos utilizados en la captura de requisitos por casos de uso son:

- Modelo de casos de uso
- Actor
- Caso de uso
- Descripción de la arquitectura – vista del modelo de casos de uso –
- Glosario
- Prototipo de interfaz de usuario

Artefacto: Modelo de casos de uso

Un modelo de casos de uso es un modelo del sistema que contiene actores, casos de uso y sus relaciones.



Artefacto: actor

El modelo de casos de uso describe lo que hace el sistema para cada tipo de usuario. Cada uno de éstos se representa mediante uno o más actores.
Los actores suelen corresponderse con trabajadores (o actores del negocio).

Artefacto: Caso de Uso

Los casos de uso son fragmentos de funcionalidad que el sistema ofrece para aportar un resultado de valor para sus actores.
Para cada caso de uso debe detallarse su **flujo de sucesos**. El flujo de sucesos puede plasmarse en forma textual y describe como interactúa el sistema con los actores cuando se lleva a cabo un caso de uso.
También pueden vincularse a un caso de uso **requisitos no funcionales** específicos del caso de uso.

Artefacto: Descripción de la arquitectura – vista del modelo de casos de uso –

La descripción de la arquitectura contiene una vista de la arquitectura del modelo de casos de uso, que representa **los casos de uso significativos** para la arquitectura.

Artefacto: glosario

Podemos utilizar un glosario para definir términos comunes importantes que los analistas utilizan para describir el sistema.

Artefacto: prototipo de interfaz de usuario

Los prototipos de interfaz de usuario nos ayudan a comprender y especificar las interacciones entre actores humanos y el sistema durante la captura de requisitos.

Trabajadores

Analizamos los trabajadores responsables de crear los artefactos mencionados.

Trabajador: analista de sistemas

Responsable de:

- modelo de casos de uso
- actores que contiene
- glosario

Trabajador: especificador de casos de uso

Asisten al analista de sistema en la descripción detallada de cada caso de uso.
Responsable de:

- caso de uso

Trabajador: diseñador de interfaz de usuario

Responsable de:

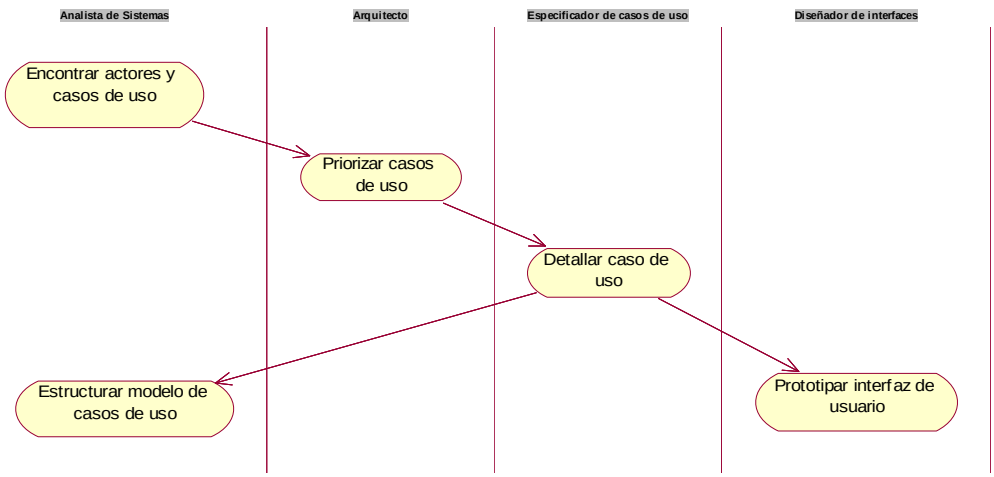
- prototipo de interfaz de usuario

Trabajador: arquitecto

Responsable de:

- descripción de la arquitectura (vista del modelo de casos de uso)

Flujo de Trabajo



Actividad: encontrar actores y casos de uso

Identificamos actores y casos de uso para:

- Delimitar el sistema de su entorno
- Esbozar quién y qué (actores) interactuarán con el sistema, y que funcionalidad (casos de uso) se espera del sistema
- Capturar y definir un glosario de términos comunes esenciales para la creación de descripciones detalladas de las funcionalidades del sistema.

Esta actividad consta de cuatro pasos:

- Encontrar los actores
- Encontrar los casos de uso
- Describir brevemente cada caso de uso
- Describir el modelo de casos de uso completo (este paso también incluye la preparación de un glosario de términos)

Estos pasos no tienen que ser ejecutados en un orden determinado, y a menudo se hacen simultáneamente.

Encontrar actores:
Depende de nuestro punto de partida.
Si partimos de un modelo del negocio, el analista puede asignar un actor a cada trabajador del negocio y un actor a cada actor del negocio (cliente) que utilizará información del sistema.
En otro caso, con o sin un modelo del dominio, el analista del sistema junto con el cliente identifica los usuarios e intenta organizarlos en categorías representadas por actores.
En ambos casos hay que identificar actores que representan sistemas externos y los actores de mantenimiento y operación del sistema.
Los actores modelan **roles**.

Encontrar casos de uso:
Cuando se parte de un modelo del negocio, se propone un caso de uso para cada rol de trabajador que utilizará información del sistema.
El método más general es la identificación a partir del análisis de cada actor por separado.

Actividad: priorizar casos de uso

El propósito de esta actividad es priorizar cuales son los casos de uso más importantes para abordar en las primeras iteraciones.
Los resultados se recogen en la vista de la arquitectura del modelo de casos de uso.
Esta vista revisada con el jefe de proyecto se utiliza como entrada al hacer la planificación de lo que debe desarrollarse dentro de una iteración.

Actividad: detallar casos de uso

El objetivo principal de detallar cada caso de uso es describir su flujo de sucesos en detalle, incluyendo como comienza, termina, e interactúan con los actores.
Para detallar los casos de uso se usan:

- Descripciones textuales

- Diagramas de transición de estados para describir los estados de los casos de uso y las transiciones entre esos estados
- Diagramas de actividad para describir transiciones entre estados con más detalle como secuencias de acciones. Los diagramas de actividad pueden describirse como la generalización de los diagramas de transición de estados
- Diagramas de interacción para describir cómo interactúa una instancia de caso de uso con la instancia de un actor. Los diagramas de interacción muestran el caso de uso y el actor o actores participantes.

Para cada caso de uso debe detallarse:

- Estado inicial como precondición
- Cómo y cuando comienza el caso de uso (primera acción a ejecutar)
- Orden en que deben ejecutarse las acciones (puede ser una secuencia numerada)
- Cómo y cuando terminan los casos de uso
- Posibles estados finales como poscondiciones
- Caminos de ejecución que no están permitidos
- Camino básico
- Caminos alternativos

Actividad: prototipar interfaz de usuario

Comenzamos con los casos de uso e intentamos discernir que se necesita de las interfaces de usuario para habilitar los casos de uso para cada actor. Hacemos un diseño lógico de la interfaz de usuario, luego creamos un modelo físico, y desarrollamos prototipos para ilustrar como pueden utilizar el sistema los usuarios para ejecutar los casos de uso.

Actividad: estructurar el modelo de casos de uso

Los casos de uso identificado son estructurados utilizando las relaciones de uso (secuencias comunes), extensiones (casos excepcionales), y generalizaciones.

7. Análisis

Introducción

Durante el análisis, analizamos los requisitos que se describieron en la captura de requisitos, refinándolos y estructurándolos. El objetivo de hacerlo es conseguir una comprensión más precisa de los requisitos y una descripción de los mismos que sea fácil de mantener y que nos ayude a estructurar el sistema entero, incluyendo su arquitectura.

Comparación del modelo de casos de uso con el modelo del análisis:

Modelo de casos de uso	Modelo de Análisis
Descrito en el lenguaje del cliente.	Descrito en el lenguaje del desarrollador.
Vista externa del sistema.	Vista interna del sistema.
Estructurado por casos de uso; proporciona la estructura a la vista externa.	Estructurado por clases y paquetes estereotipados; proporciona la estructura a la vista interna.
Utilizado fundamentalmente como contrato entre el cliente y los desarrolladores sobre qué debería y qué no debería hacer el sistema.	Utilizado fundamentalmente por los desarrolladores para comprender cómo deberá darse forma al sistema, es decir, cómo debería ser diseñado e implementado.
Puede contener redundancias e inconsistencias entre requisitos.	No debería contener redundancias ni inconsistencias entre requisitos.
Captura la funcionalidad del sistema, incluida la funcionalidad significativa para la arquitectura.	Esboza cómo llevar a cabo la funcionalidad dentro del sistema, incluida la funcionalidad significativa para la arquitectura; sirve como una primera aprox. al diseño.
Define casos de uso que se analizarán con más profundidad en el modelo de análisis.	Define realizaciones de caso de uso, y cada una de ellas representa el análisis de un caso de uso del modelo de casos de uso.

El lenguaje que utilizamos en el análisis se basa en un modelo de objetos conceptual, que llamamos **modelo de análisis**. El modelo de análisis nos ayuda a **refinar** los requisitos.
Analizar los requisitos en la forma de un modelo de análisis es importante por varios motivos:

- Un modelo de análisis ofrece una especificación más precisa de los requisitos que la que tenemos como resultado de la captura de requisitos, incluyendo al modelo de casos de uso.
- Un modelo de análisis se describe utilizando el lenguaje de los desarrolladores, y se puede por tanto intruducir un mayor formalismo y ser utilizado para razonar sobre los funcionamientos internos del sistema.
- Un modelo de análisis estructura los requisitos de un modo que facilita su comprensión, su preparación, su modificación, y en general, su mantenimiento.
- Un modelo de análisis puede considerarse como una primera aproximación al modelo de diseño (aunque es un modelo por sí mismo), y es por tanto una entrada fundamental cuando se da forma al sistema en el diseño y en la implementación.

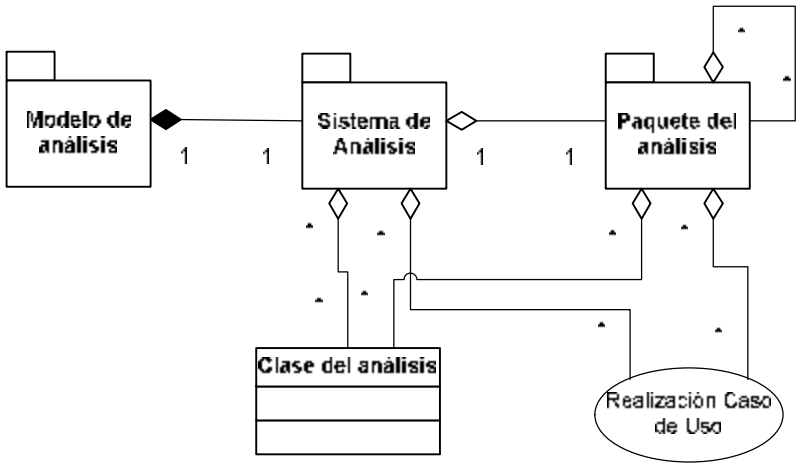
Trabajadores y Artefactos

Trabajador	Responsable de (artefacto)
Arquitecto	Modelo de Análisis
	Descripción de la arquitectura
Ingeniero de Casos de Uso	Realización de casos de usos – Análisis -

Ingeniero de Componentes	Clases del Análisis
	Paquete del análisis

Artefactos

Artefacto: Modelo del análisis



Compuesto por un sistema de análisis que es el paquete de más alto nivel, el cual se compone a su vez de otros paquetes y clases de análisis y realizaciones de casos de uso.

Artefacto: clase del análisis

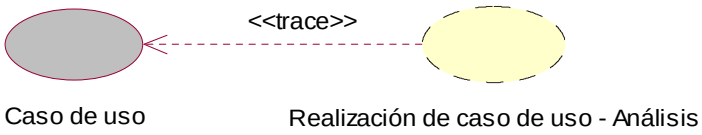
Una clase de análisis represneta una abstracción de una o varias clases y/o subsistemas del diseño. Características:

- Se centra en el tratamiento de requisitos funcionales y pospone los no funcionales para el diseño.
- Es más “conceptual”.
- Raramente define una interfaz en términos de operaciones y sus signaturas. Su comportamiento se define mediante “responsabilidades” en un nivel más alto y menos formal. Una responsabilidad es una descripción textual de un conjunto cohesivo del comportamiento de una clase.
- Define atributos en un nivel tambien conceptual y reconocibles en el dominio del problema, mientras que en el diseño los atributos se ajustan a tipos del lenguaje de programación.
- Las relaciones entre clases del análisis también son más conceptuales. Por ejemplo no se da importancia a la navegación de la relación.
- Las clases de análisis siempre encajan en alguno de los estereotipos básicos: de interfaz, de control, o entidad.

- Clases entidad: se derivan de las clases entidad del negocio (o dominio).
- Clases de control: encapsulan el control de casos de uso, o cálculos complejos.
- Clases de interfaz: modelan la interacción entre actores y el sistema.

Artefacto: realización caso de uso análisis

Una realización de caso de uso – análisis es una colaboración dentro del modelo de análisis que describe cómo se lleva a cabo y se ejecuta un caso de uso determinado en términos de las clases del análisis y sus objetos del análisis en interacción.
Una realización de caso de uso análisis posee una *descripción textual del flujo de sucesos, diagramas de clases participantes*, y *diagramas de interacción* que muestran la realización de un flujo o escenario particular del caso de uso en término de objetos del análisis.



Caso de uso Realización de caso de uso - Análisis

Artefacto: paquete del análisis

Los paquetes del análisis proporcionan un medio para organizar los artefactos del modelo de análisis en piezas manejables. Un paquete de análisis puede constar de clases de análisis, de realizaciones de casos de uso, y de otros paquetes del análisis recursivamente.
Los paquetes del análisis son particionamientos funcionales del sistema basados en el dominio del problema y debería ser reconocibles por las personas con conocimiento del dominio.
Los paquetes del análisis probablemente se convertirán en subsistemas en las dos capas de aplicación superiores del modelo de diseño, o se distribuirán entre ellos.

Paquete de servicio

Un servicio representa un conjunto coherente de acciones relacionadas funcionalmente –un paquete de funcionalidad- que se utiliza en varios casos de uso. Un cliente de un sistema normalmente compra una combinación de servicios para ofrecer a sus usuarios los casos de uso necesario. Un servicio es indivisible en el sentido de que el sistema necesita ofrecerlo o todo entero o nada en absoluto.
Los casos de uso atraviesan los servicios, es decir, un caso de uso requiere acciones de varios servicios.
En RUP, el concepto de servicio está soportado por los paquetes de servicio.
Los paquetes de servicio se utilizan en el nivel más bajo de la jerarquía (de agregación) de paquetes de análisis para estructurar el sistema de acuerdo a los servicios que proporciona.
Podemos observar lo siguiente acerca de los paquetes de servicio:

- Un paquete de servicios contiene un conjunto de clases relacionadas funcionalmente.
- Un paquete de servicios es indivisible.
- Para llevar a cabo un caso de uso puede que participen más de un paquete de servicios.
- Un paquete de servicios puede depender de otro paquete de servicios.

- Un paquete de servicios normalmente es relevante para un pequeño grupo de actores.
- Un paquete de servicios puede gestionarse como una unidad de distribución independiente. Puede representar una funcionalidad "adicional" del sistema.
- Los paquetes de servicio pueden ser mutuamente excluyentes, o pueden representar diferentes variantes del mismo servicio.
- Los paquetes de servicio constituyen la entrada fundamental para las actividades de diseño e implementación subsiguientes, dado que ayudarán a estructruar los modelos de diseño e implementación en términos de subsistemas de servicio.

Artefacto: descripción de la arquitectura (vista del modelo de análisis)

Los siguientes artefactos del modelo de análisis se consideran significativos para la arquitectura:

- Descomposición del modelo de análisis en paquetes de análisis y sus dependencias. Esta descomposición suele tener su efecto en los subsistemas de las capas superiores durante el diseño e implementación.
- Las clases fundamentales del análisis.
- Realizaciones de casos de uso que describen funcionalidades importantes y críticas, probablemente las correspondientes a los casos de uso que aparecen en la vista de la arquitectura del modelo de casos de uso.

Trabajadores

Trabajador: Arquitecto

Es responsable de la integridad del modelo de análisis, garantizando que sea correcto, consistente, y legible como un todo.
El arquitecto es responsable de:

- la descripción de la arquitectura
- modelo del análisis

Trabajador: Ingeniero de casos de uso

Es responsable de la integridad de una o más realizaciones de caso de uso, garantizando que cumplen los requisitos que recaen sobre ellos.
El ing.de casos de uso es responsable de:

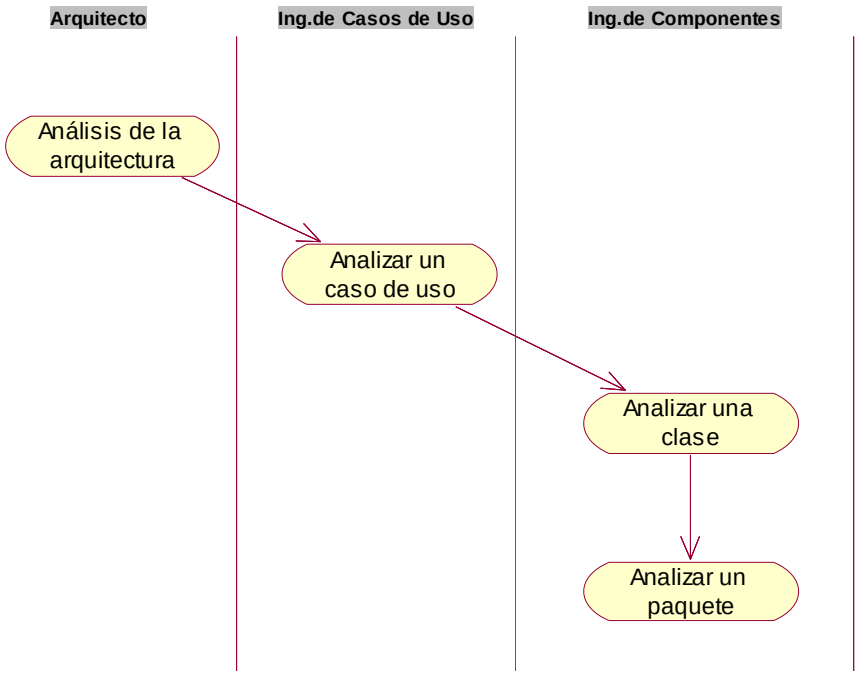
- realización de casos de uso – análisis

Trabajador: Ingeniero de componentes

Define y mantiene las responsabilidades, atributos, relaciones, y requisitos especiales de una o varias clases del análisis asegurándose de que cada clase del análisis cumple los requisitos que se esperan de ella de acuerdo a las realizaciones de caso de uso en que participan.
También mantiene la integridad de uno o varios paquetes del análisis.
El ingeniero de componentes es responsable de:

- clase del análisis
- paquete del análisis

Flujo de Trabajo



Actividad: análisis de la arquitectura

El propósito de análisis de la arquitectura es esbozar el modelo de análisis y la arquitectura mediante la identificación de paquetes del análisis, clases del análisis evidentes, y requisitos especiales comunes.

Identificación de paquetes de análisis

Los paquetes proporcionan un medio para organizar el modelo de análisis en piezas más pequeñas y manejables.
Pueden identificarse inicialmente como forma de dividir el análisis o encontrarse a medida que se avanza en el análisis.
Una identificación inicial se hace de manera natural basándonos en los requisitos funcionales y en el dominio de problema, agrupando un cierto número de casos de uso en un paquete concreto, y realizando la funcionalidad correspondiente dentro de dicho paquete. Algunos criterios para agrupar casos de uso son:

- Casos de uso para dar soporte a un determinado proceso de negocio.
- Casos de uso para dar soporte a un determinado actor del sistema.
- Casos de uso relacionados mediante relaciones de generalización y de extensión.

Cuando dos paquetes necesitan compartir una misma clase, es conveniente hubicar dicha clase en su propio paquete.

Identificación de paquetes de servicio

La identificación de paquetes de servicio se suele hacer cuando el trabajo de análisis está avanzado, cuando se comprenden bien los requisitos funcionales, y existen la mayoría de las clases del análisis.
Para identificar paquetes de servicio debemos:

- Identificar un paquete de servicio por cada servicio opcional. El paquete de servicio será una unidad de compra. (por ej. enviar avisos a clientes)

- Identificar un paquete de servicio por cada servicio que podría hacerse opcional, incluso aunque todos los clientes siempre lo quieran.

Definición de dependencias entre paquetes del análisis

Deben definirse dependencias entre los paquetes del análisis si sus contenidos están relacionados. La dirección de la dependencia debería ser la misma (navegabilidad) dirección de la relación.
Buscamos definir paquete que sean débilmente acoplados y altamente cohesivos con respecto a las clases que contienen.
Para hacer más claras las dependencias puede ser útil estratificar el modelo de análisis haciendo que los paquetes específicos de la aplicación queden en una capa de nivel superior y los paquetes generales queden en una capa inferior.

Identificación de clases de entidad obvias

Pueden identificarse una lista de clases entidad candidatas basado en las clases del dominio o las entidades del negocio.
Sin embargo la mayoría de las clases se identificarán al crear las realizaciones de casos de uso. Por lo cual en esta etapa es conveniente con un esbozo inicial de las clases significativas para la arquitectura.

Identificación de requisitos especiales

Un requisito especial es un requisito que aparece durante el análisis y que es importante anotar de forma que pueda ser tratado adecuadamente en las subsiguientes actividades de diseño e implementación.
Como ejemplos podemos citar restricciones sobre:

- persistencia
- distribución y concurrencia
- características de seguridad
- tolerancia a fallos
- gestión de transacciones

Actividad: analizar un caso de uso

Analizamos un caso de uso para:

- Identificar las clases del análisis cuyos objetos son necesarios para llevar a cabo el flujo de suceso del caso de uso.
- Distribuir el comportamiento del caso de uso entre objetos del análisis que interactúan.
- Capturar requisitos especiales sobre la realización del caso de uso.

Identificación de clases del análisis

Buscamos clases de entidad, control, e interfaz y esbozamos sus nombres, responsabilidades, atributos, y relaciones.
Podemos utilizar las siguientes guías para identificar clases:

- Identificar clases entidad a partir de considerarse que información debe utilizarse y manipularse para realizar el caso de uso.

- Identificar una clase de interfaz para cada actor humano, y dejar que esta clase represente la ventana principal de la interfaz de usuario con la cual interactúa el actor.
- Identificar una clase de interfaz primitiva para cada clase de entidad que hayamos encontrado anteriormente. Estas clases representan objetos lógicos con los que interactúa el actor en la interfaz de usuario.
- Identificar una clase de interfaz central para cada actor que sea un sistema externo.
- Identificar una clase de control responsable del tratamiento del control y de la coordinación de la realización del caso de uso, y después refinar esta clase de control de acuerdo a los requisitos del caso de uso.

Descripción de las interacciones entre objetos del análisis

Se utiliza un diagrama de colaboración para describir como interactúan los objetos encontrados para realizar el caso de uso. Podemos observar lo siguiente:

- Un caso de uso se inicia mediante un mensaje proveniente de una instancia de un actor.
- Cada clase identificada en el paso anterior debe tener una instancia en esta colaboración.
- Los mensajes no se asocian con operaciones, ya que las clases de análisis se definen en término de responsabilidades no en operaciones atómicas.
- Los enlaces del diagrama son instancias de las asociaciones entre clases del análisis.

Captura de requisitos especiales

Recogemos todos los requisitos adicionales inherentes al caso de uso que se está tratando.

Actividad: analizar una clase

Los objetivos de analizar una clase son:

- Identificar y mantener las responsabilidades de la clase, basadas en su papel en las realizaciones de casos de uso.
- Identificar atributos y relaciones de la clase.
- Capturar requisitos especiales sobre la realización de la clase.

Actividad: analizar un paquete

Los objetivos de analizar una clase son:

- Garantizar que el paquete es tan independiente de otros como sea posible.
- Garantizar que el paquete del análisis cumple su objetivo de realizar algunas clases del dominio o casos de uso.
- Describir las dependencias de forma que pueda estimarse el efecto de los cambios futuros.

8. Diseño

Introducción

Durante el diseño modelamos el sistema y su arquitectura para que soporte los requisitos funcionales y no funcionales. Una entrada esencial al diseño es el modelo de análisis.

Comparación modelo del análisis – modelo del diseño

Modelo de Análisis	Modelo de Diseño
Modelo conceptual.	Modelo físico (implementación)
Genérico respecto al diseño (aplicable a varios diseños)	Específico para una implementación
Tres estereotipos: entidad, control, interface.	Cualquier nro. de estereotipos físicos.
Menos formal.	Mas formal.
Menos caro de desarrollar	Más caro.
Menos capas.	Más capas.
Dinámico (no muy centrado en la secuencia)	Dinámico (muy centrado en la secuencia)
Creado principalmente como trabajo manual	Creado fundamentalmente como "programación visual" en ing.de ida y vuelta.
Puede no mantenerse todo el ciclo de vida.	Debe ser mantenido todo el ciclo de vida.

El diseño es el centro de atención al final de la fase de elaboración y comienzo de las iteraciones de construcción.

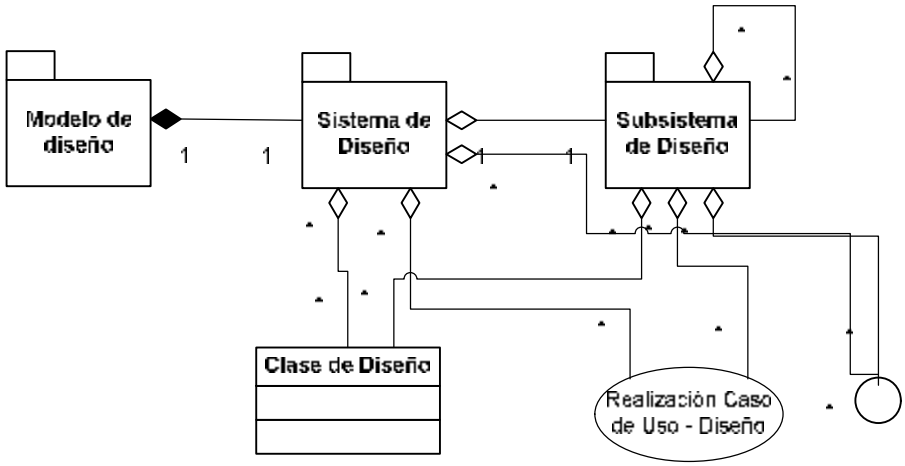
Trabajadores y Artefactos

Trabajador	Responsable de (artefacto)
Arquitecto	Modelo de diseño
	Modelo de despliegue
	Descripción de la arquitectura
Ingeniero de Casos de Uso	Realización de casos de usos – Diseño -
Ingeniero de Componentes	Clases del diseño
	Subsistema de Diseño
	Interfaz

Artefactos

Artefacto: Modelo del Diseño

El modelo de diseño es un modelo de objetos que describe la realización física de los casos de uso centrándose en los requisitos funcionales como en los no funcionales.
Las abstracciones del modelo de diseño tienen una correspondencia directa con los elementos físicos del ambiente de implementación.



Artefacto: Clase del Diseño

Una clase de diseño es una abstracción sin costuras con una clase o construcción similar en la implementación del sistema.

- Se especifica en el lenguaje de implementación (Java, C#, etc).
- Se especifica visibilidad de atributos y operaciones.
- Se implementan las relaciones de generalización via herencia del lenguaje de programación.
- Se implementan las relaciones de asociación y agregación via atributos de las clases que implementan las referencias.
- Se especifican los métodos con la sintaxis del lenguaje de programación.
- Se usan estereotipos que se corresponden con el lenguaje de programación, por ejemplo en VB se puede utilizar *<<class module>>,* *<<form>>*, etc.

Artefacto: realización de caso de uso-diseño

Es una colaboración en término de clases de diseño que describe como se realiza un caso de uso específico. Una realización de caso de uso diseño, tiene una traza directa a la correspondiente realización del caso de uso análisis.
Una realización de caso de uso-diseño se describe utilizando:

- Diagramas de clases para mostrar los objetos participantes en la realización.
- Diagrama de interacción, particularmente diagramas de secuencia, enfatizando en la secuencia de mensajes.
- Flujos de sucesos-diseño, descripción textual que explica y complementa los diagramas y sus etiquetas.

Artefacto: subsistema de diseño

Los subsistemas de diseño son una forma de organizar los artefactos del modelo de diseño en piezas más manejables. Un subsistema puede constar de clases de diseño, realizaciones de caso de uso, interfaces, y otros subsistemas.

Un subsistema puede proporcionar interfaces que representan la funcionalidad que exportan en término de operaciones.
Los subsistemas pueden representar separación de un sistema grande en subsistemas que pueden desarrollarse por equipos separados.
Las dos capas de aplicación de nivel más alto suelen tener trazas directas hacia paquetes y/o clases del análisis.
Los subsistemas pueden representar productos software reutilizados.
Los subsistemas pueden representar sistemas heredados encapsulándolos.

Subsistemas de Servicio: Los subsistemas de servicio diseño se usan en un nivel inferior de la jerarquía de subsistemas. La identificación de subsistemas de servicio se basa en los **paquetes** de servicio del modelo de análisis, y normalmente existe una traza uno a uno.
Un subsistema de servicio ofrece servicios en término de interfaces y operaciones.
Un subsistema de servicio suele dar lugar a un componente ejecutable en la implementación.

Artefacto: interfaz

Las interfaces se utilizan para especificar las operaciones que proporcionan las clases y los subsistemas del diseño.
Se dice que una clase de diseño o un subsistema de diseño “realizan” o implementan una interfaz.
Las interfaces constituyen una forma de separar la especificación de la funcionalidad en términos de operaciones de sus implementaciones en términos de métodos.

Artefacto: descripción de la arquitectura (vista del modelo de diseño)

Se consideran significativos para la arquitectura los siguientes artefactos del modelo de diseño:

- La descomposición del modelo de diseño en subsistemas, sus interfaces, y las dependencias entre ellos.
- Clases de diseño fundamentales como clases activas y clases centrales.
- Realizaciones de caso de uso-diseño que describan alguna funcionalidad importante y crítica.

Artefacto: modelo de despliegue

El modelo de despliegue es un modelo de objetos que describe la distribución física del sistema en términos de cómo se distribuye la funcionalidad entre los nodos de cómputo.

Artefacto: descripción de la arquitectura (vista del modelo de despliegue)

La descripción de la arquitectura contiene una vista de la arquitectura del modelo de despliegue que muestra sus artefactos relevantes para la arquitectura.

Trabajadores

Trabajador: arquitecto

Responsable del modelo de diseño, modelo de despliegue, y descripción de la arquitectura.

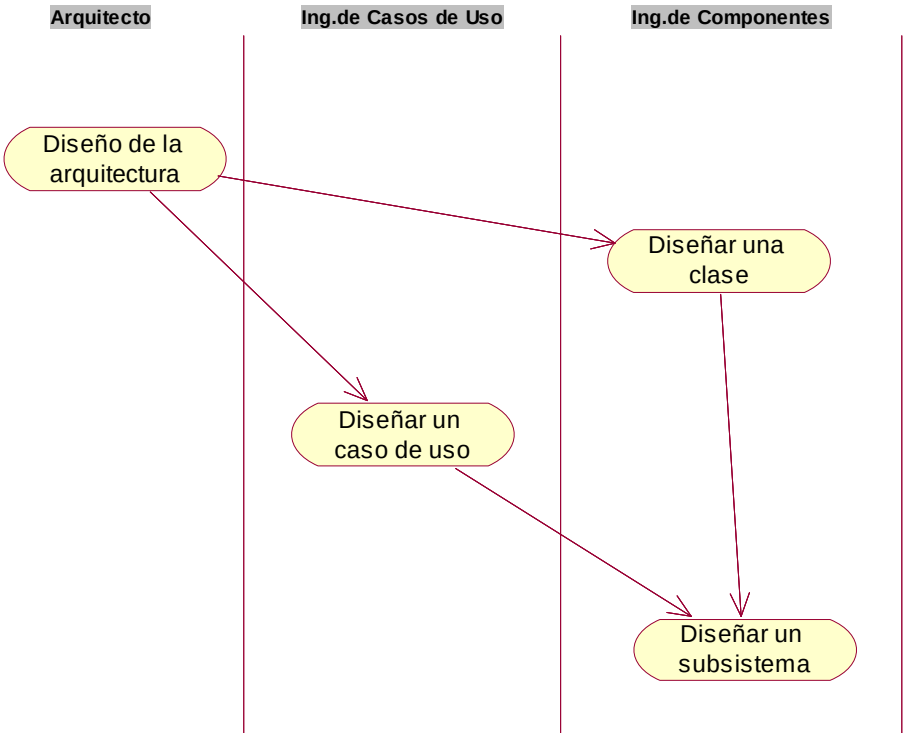
Trabajador: ingeniero de casos de uso

Responsable del realización de caso de uso – diseño.

Trabajador: ingeniero de componentes

Responsable de Clases de diseño, subsistema de diseño, interfaz.

Flujo de Trabajo



Actividad: diseño de la arquitectura

El objetivo del diseño de la arquitectura es esbozar los modelos de diseño y despliegue identificando:

- Nodos y configuraciones de red.
- Subsistemas e interfaces
- Clases de diseño significativas para la arquitectura como las clases activas (procesos).
- Mecanismos de diseño genéricos que tratan requisitos comunes.

Nodos y configuraciones de red

Entre los aspectos de configuraciones de red podemos resaltar:

- ¿Qué nodos se necesitan, que capacidad deben tener?
- ¿Qué tipo de conexiones debe haber entre los nodos, que protocolos de comunicaciones?
- ¿Qué características deben tener los nodos, ancho de banda, disponibilidad, calidad, etc?
- ¿procesos redundantes, capacidad de fallos, etc?

Identificación de subsistemas y de sus interfaces

Los subsistemas constituyen un medio para organizar el modelo de diseño en piezas manejables.
Pueden identificarse inicialmente como forma de dividir el trabajo de diseño, o bien pueden irse encontrando a medida que el modelo de diselo evoluciona.

Subsistemas de aplicación: Son las capas superiores de la aplicación y pueden derivarse directamente de los paquetes de análisis. Se distingue capa específica y general de la aplicación.

Subsistemas intermedios y de software del sistema: Constituyen los cimientos de un sistema. Son sistemas operativos, SGBD, software de comunicaciones, tecnologias de distribución de objetos, kits de construcción de GUIs, gestores de transacciones, etc.

Actividad: diseño de un caso de uso

Los objetivos del diseño de un caso de uso son:

- Identificar clases y/o subsistemas necesarios para llevar a cabo el caso de uso
- Distribuir el comportamiento del caso de uso entre los objetos del diseño que interactúan y/o entre los subsistemas participantes.
- Definir los requisitos sobre las operaciones de las clases del diseño y/o sobre los subsistemas y sus interfaces.
- Capturar los requisitos de implementación del caso de uso.

Actividad: diseño de una clase

Para una clase implica definir:

- sus operaciones
- sus atributos
- sus relaciones
- sus métodos (que realizan sus operaciones)
- su ciclo de vida (máquina de estados)
- sus dependencias con cualquier mecanismo de diseño generico
- los requisitos relevantes a su implementación
- la correcta realizacion de cualquier interfaz requerida

Esbozar la clase del diseño:

- Diseñar clase interfaz: depende de la tecnología específica que se use (VB, Java, etc)

- Diseñar clases entidad: aspectos de persistencia, a menudo implica uso de tecnologías BD.
- Diseñar clases de control: encapsulan secuencias, coordinación de otros objetos, y aveces lógica del negocio. Se deben tener en cuenta: aspectos de distribución en nodos de red, aspectos de rendimiento, aspectos de transacción.

Identificar Operaciones:
Identificamos las operaciones y las describimos utilizando el lenguaje de programación que se usará. Entradas importantes son:

- Responsabilidades de las clases de análisis que tienen traza con la clase de diseño
- Requisitos especiales de la clase de análisis que tienen traza
- Interfaces que la clase de diseño necesita proporcionar
- Realizaciones de caso de uso diseño en las que la clase participa

Identificar Atributos:
Identificamos atributos requeridos por la clase y los describimos utilizando el lenguaje de programación que se usará.

Identificar Asociaciones y agregaciones:
Los ingenieros de componentes estudian la transmisión de mensajes en los diagramas de secuencia para determinar que asociaciones son necesarias.
Aspectos a tener en cuenta:

- Asociaciones y agregaciones en las clases de análisis correspondientes
- Los nombres de roles de asociación pueden llegar a ser atributos de la clase de diseño que referencia a la otra clase.
- Una asociación de clases puede llegar a ser una nueva clase.
- Se debe considerar la navegabilidad (dirección de los mensajes).

Identificar generalizaciones:
Las generalizaciones deben implementarse utilizando herencia si el lenguaje lo permite. Si el lenguaje no lo admite debe utilizarse asociación / agregación para proporcionar la delegación desde los objetos de clases más específicas a objetos de clases más generales.

Describir los métodos:
Pueden describirse los algoritmos de los métodos utilizando lenguaje natural, pseudocódigo, o el lenguaje de programación específico. Sin embargo esto suele diferirse hasta la implementación de la clase.

Describir estados:
Se utilizan diagramas de estado para describir los estados de los objetos estado dependientes.

Actividad: diseño de un subsistema

Los objetivos del diseño de un subsistema son:

- Garantizar que el subsistema es lo más independiente de los otros subsistemas.
- Garantizar que el susbsistema proporciona las interfaces correctas.
- Garantizar que el subsistema cumple su propósito de ofrecer una realización correcta de las operaciones que se definen en las interfaces.

9. Implementación

Introducción

En la implementación empezamos con el resultado del diseño e implementamos el sistema en término de componentes, es decir, ficheros de código fuente, scripts, ficheros de código binario, ejecutables, y similares.

La implementación es el centro durante las iteraciones de construcción, aunque también se lleva a cabo trabajo de implementación durante la fase de elaboración, para crear la línea base ejecutable de la arquitectura, y durante la fase de transición para tratar defectos tardíos.

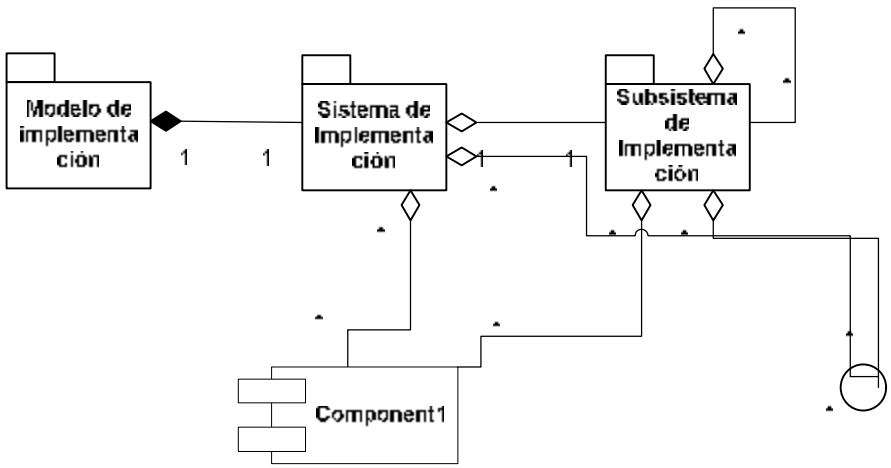
Trabajadores y Artefactos

Trabajador	Responsable de (artefacto)
Arquitecto	Modelo de implementación
	Modelo de despliegue
	Descripción de la arquitectura
Integrador de sistema	Integración de sistema
Ingeniero de Componentes	Componente
	Implementación de subsistema
	Interfaz

Artefactos

Artefacto: Modelo de Implementación

El modelo de diseño es un modelo de objetos que describe la realización física de los elementos del modelo de diseño.



Artefacto: componente

Un componente es el empaquetamiento físico de los elementos de un modelo, como son las clases en el modelo de diseño. Algunos estereotipos típicos son:
<<executable>> programa ejecutable en un nodo
<<file>> fichero que contiene código fuente o datos
<<library>> librería estática o dinámica
<<table>> es una tabla de base de datos
<<document>> es un documento

Los componentes tienen relaciones de traza con los elementos que implementan.
Es normal que un componente implemente varios elementos, por ejemplo varias clases.

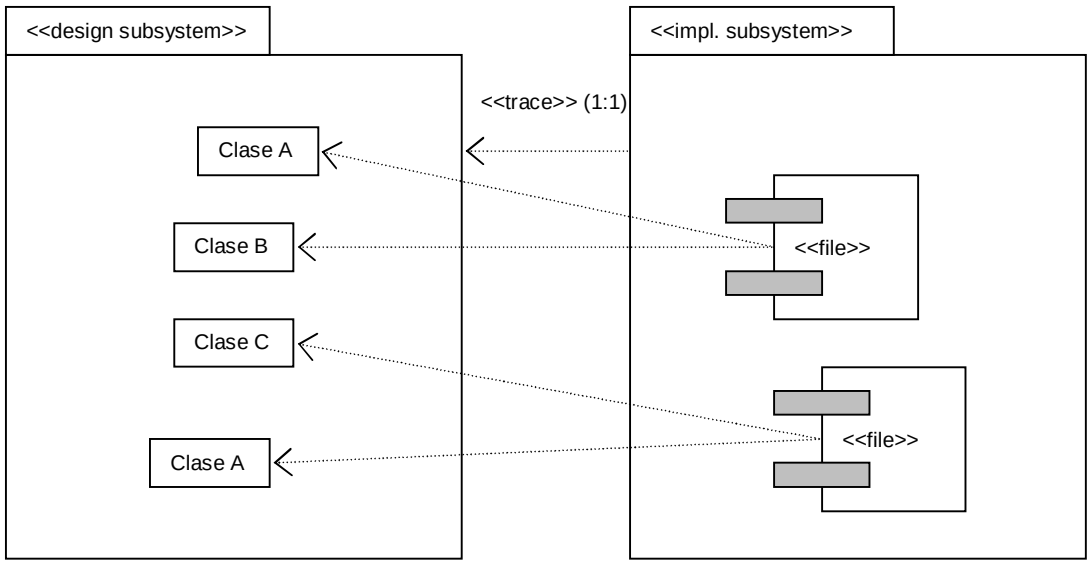
Stubs
Un Stub es un componente con una implementación esquelética o de propósito especial que puede ser utilizado para desarrollar o probar otro componente que depende del stub.

Artefacto: subsistema de implementación

Los subsistemas de implementación proporcionan una forma de organizar los artefactos del modelo de implementación en trozos más manejables.
Un subsistema de implementación se manifiesta a través de un mecanismo de empaquetamiento concreto en un entorno de implementación determinado, como:

- Un paquete en Java
- Un proyecto en VB.
- Un directorio de ficheros en un proyecto de C++
- Un paquete en una herramienta de modelado como Rational Rose.

Los subsistemas de implementación están muy relacionados con los subsistemas de diseño con los cuales tienen una traza uno a uno (isomórfica).



Artefacto: interfaz

En el modelo de implementación las interfaces pueden ser utilizadas para especificar las operaciones implementadas por componentes y subsistemas de implementación.

Artefacto: descripción de la arquitectura

La descripción de la arquitectura tiene la visión de la arquitectura del modelo de implementación.
Los siguientes artefactos son considerados arquitectónicamente significativos:

- Descomposición del modelo de implementación en subsistemas, sus interfaces, y dependencias entre ellos.
- Componentes clave que siguen la traza de las clases de diseño significativas arquitectónicamente.

Artefacto: plan de integración

El sistema se desarrolla incrementalmente a pasos manejables. Cada paso de construcción debe ser sometido a pruebas de integración.
Para prepararse ante el fallo de una construcción se lleva un control de versiones para poder volver atrás a una construcción anterior.

Un plan de integración de construcciones describe la secuencia de construcciones necesarias en una iteración. Un plan de este tipo describe lo siguiente para cada construcción:

- La funcionalidad que se espera sea implementada en dicha construcción, consiste en una lista de casos de uso a implementar.
- Las partes del modelo de implementación que están afectadas por la construcción (lista de subsistemas y componentes necesarios para implementar la funcionalidad).

Trabajadores

Trabajador: arquitecto

Durante la fase de implementación, el arquitecto es responsable de la integridad del modelo de implementación y asegura que el modelo como un todo es correcto, completo, y legible.
El arquitecto es responsable del modelo de implementación, el modelo de despliegue, y la descripción de la arquitectura.

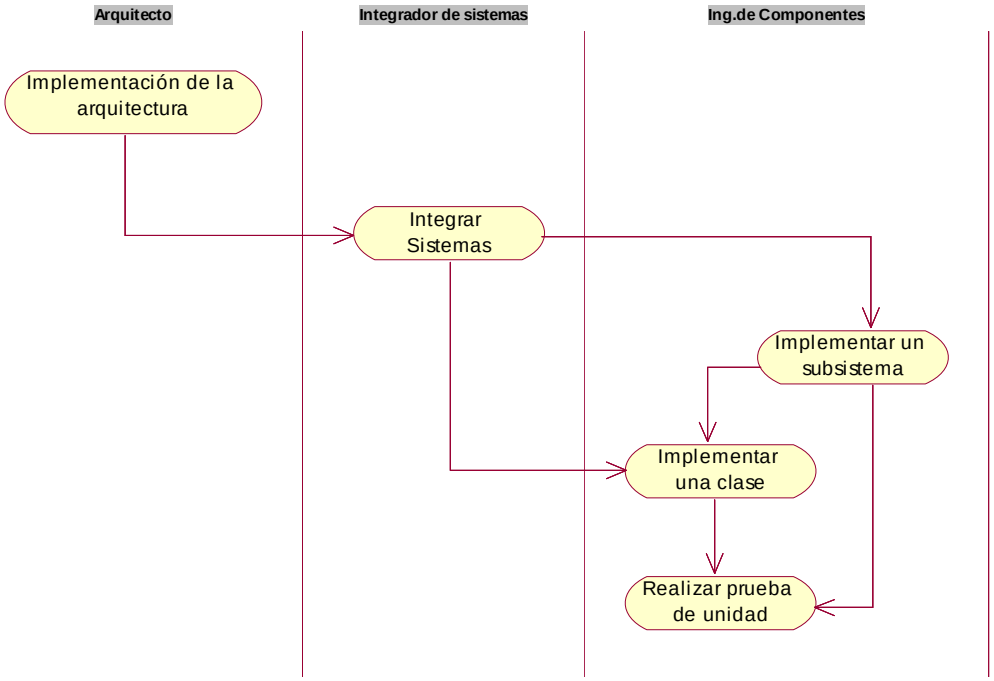
Trabajador: ingeniero de componentes

Define y mantiene el código fuente de uno o varios componentes, garantizando que el componente implementa la funcionalidad correcta.
A menudo también mantiene la integridad de uno o varios subsistemas de implementación.

Trabajador: integrador de sistemas

Es responsable del plan de integración de construcciones.

Flujo de Trabajo



Actividad: implementación de la arquitectura

El propósito de la implementación de la arquitectura es esbozar el modelo de implementación y su arquitectura mediante:

- Identificación de componentes significativos arquitectónicamente tales como componentes ejecutables.
- La asignación de componentes a los nodos en las configuraciones de redes relevantes.

Para esto consideramos las clases activas encontradas durante el diseño y asignamos un componente ejecutable a cada clase activa.

Actividad: integrar el sistema

Los objetivos de la integración son:

- Crear un plan de integración de construcciones
- Integrar cada construcción antes de que sea sometida a pruebas de integración.

Actividad: implementar un subsistema

El propósito de implementar un subsistema es el de asegurar que un subsistema cumpla su papel en cada construcción.

Actividad: implementar una clase

El propósito de implementar una clase es implementar una clase de diseño en un componente fichero. Esto incluye:

- Esbozo de un componente fichero que contendrá el código.
- Generación del código fuente a partir de la clase de diseño y de las relaciones en que participa.
- Implementación de las operaciones de la clase de diseño en forma de métodos.
- Comprobación de que el componente proporciona las mismas interfaces que la clase de diseño.

Actividad: realizar prueba de unidad

El propósito de realizar la prueba de unidad es probar los componentes implementados como unidades individuales. Existen dos tipos de prueba:

- Prueba de especificación, o prueba de “caja negra”, que verifica el comportamiento de la unidad observable externamente.
- Prueba de estructura o prueba de “caja blanca”, que verifica la implementación interna de la unidad.

10. Prueba

Introducción

Los objetivos de la prueba son:

- Planificar las pruebas necesarias en cada iteración, incluyendo las pruebas de integración y las pruebas de sistema.
- Diseñar e implementar pruebas creando los casos de prueba (especifican qué probar), procedimientos de prueba (especifican cómo realizar las pruebas), creando componentes de prueba para automatizar las pruebas.
- Realizar las pruebas.

Trabajadores y Artefactos

Trabajador	Responsable de (artefacto)
Diseñador de Pruebas	Modelo de pruebas
	Casos de Prueba
	Procedimientos de prueba
	Evaluación de pruebas
	Plan de pruebas
Ingeniero de Componentes	Componente de pruebas
Ingeniero de Pruebas de Integración	Defecto
Ingeniero de Pruebas de Sistema	Defecto

Artefactos

Artefacto: Modelo de pruebas

Describe como se prueban los componentes en el modelo de implementación. El modelo de pruebas es una colección de *casos de prueba*, *procedimientos de prueba*, y *componentes de prueba*.

Artefacto: Caso de prueba

Un caso de prueba especifica una forma de probar el sistema, incluyendo la entrada o resultado con la que se ha de probar y las condiciones bajo las que ha de probarse.

Artefacto: Procedimiento de prueba

Un procedimiento de prueba especifica cómo realizar uno o varios casos de prueba o partes de estos.

Artefacto: Componente de prueba

Un componente de prueba automatiza uno o varios procedimientos de prueba o partes de ellos.

Artefacto: Plan de prueba

El plan de prueba describe las estrategias, recursos y planificación de la prueba. La estrategia incluye definición de tipo de pruebas a realizar por iteración, sus objetivos, nivel de cobertura y código necesario.

Artefacto: Defecto

Un defecto es una anomalía del sistema. Un defecto puede ser utilizado para localizar cualquier cosa que los desarrolladores necesitan registrar como síntoma de un problema.

Artefacto: Evaluación de prueba

Una evaluación de prueba es una evaluación de los resultados de la prueba.

Trabajadores

Trabajador: diseñador de pruebas

Un diseñador de pruebas es responsable de la integridad del modelo de pruebas, asegurando que el modelo cumple con su propósito.
Planean las pruebas.
Seleccionan y describen los casos de prueba y procedimientos de prueba.

Trabajador: ingeniero de componentes

Son responsables de los componentes de prueba que automatizan algunos de los procedimientos de prueba.

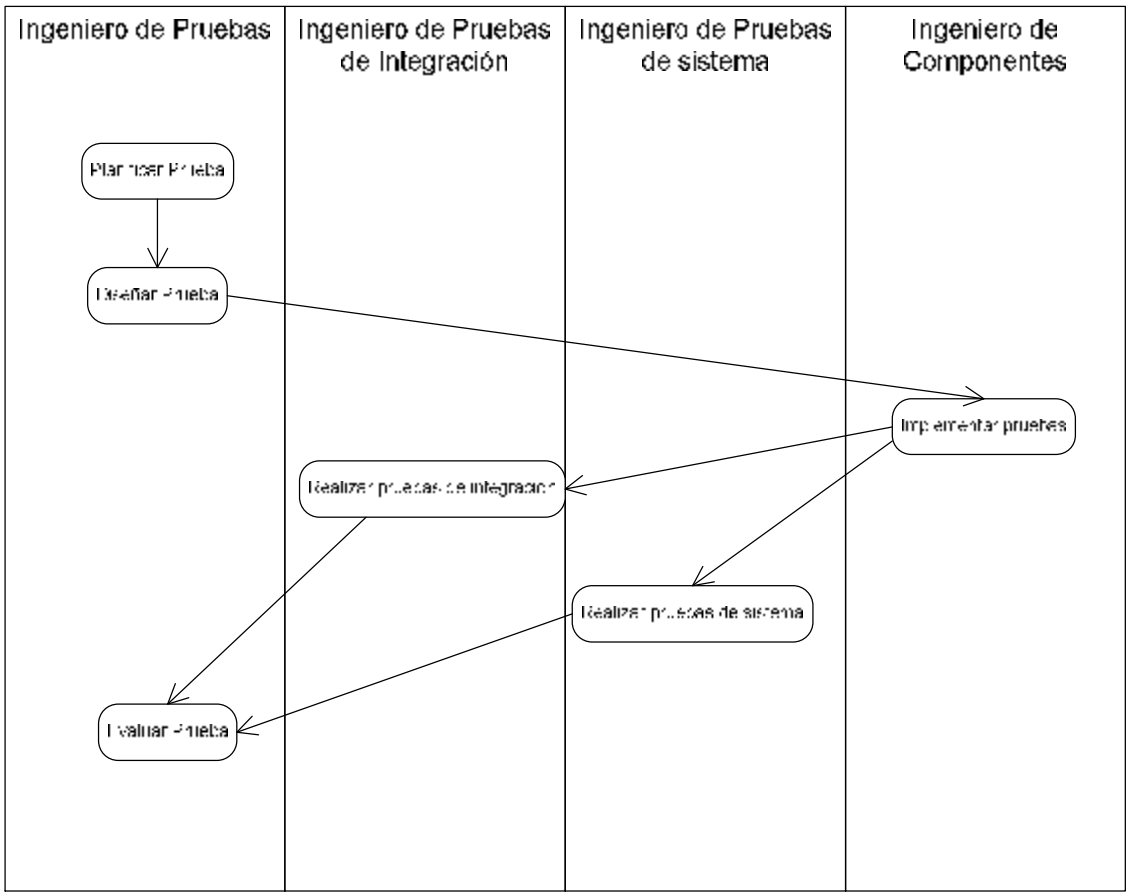
Trabajador: ingeniero de pruebas de integración

Son los responsables de realizar las pruebas de integración que se necesitan para cada construcción producida en el flujo de implementación.
Documentan los defectos que aparecen como resultados de las pruebas de integración.

Trabajador: ingeniero de pruebas de sistema

Un ingeniero de pruebas de sistema es responsable de realizar las pruebas de sistema necesarias sobre una construcción que muestra el resultado de una iteración completa.
Las pruebas de sistema se llevan a cabo para verificar interacciones entre los actores y el sistema.

Flujo de trabajo



Actividad: planificar prueba

- Describir una estrategia de la prueba
- Estimar requisitos para la prueba, recursos humanos y sistemas necesarios
- Planificar esfuerzo de la prueba

Actividad: diseñar la prueba

- Identificar y describir los casos de prueba para cada construcción
- Identificar y estructurar los procedimientos de prueba especificando como realizar los casos de prueba.

Actividad: implementar la prueba

- Automatizar los procedimientos de prueba creando componentes de prueba si esto es posible.

Actividad: realizar pruebas de integración

- Realizar las pruebas de integración relevantes ejecutando los procedimientos o componentes de prueba correspondientes.
- Comparar los resultados de las pruebas con los resultados esperados e investigar resultados no esperados.
- Informar defectos a los ingenieros de componentes responsables de los componentes que registran fallas.

- Informar los defectos a los diseñadores de pruebas, quienes usarán los defectos para evaluar los resultados de las pruebas.

Actividad: realizar prueba del sistema

Una vez finalizadas las pruebas de integración se realizan las pruebas de sistema de forma similar.

Actividad: evaluar la prueba

Se comparan resultados de la prueba con resultados esperados. Para esto se utilizan *métricas*:

- Compleción de la prueba: indica el porcentaje de casos de prueba que han sido ejecutados y el porcentaje de código que ha sido probado.
- Fiabilidad: Se basa en el análisis de las tendencias den los defectos detectados y en las tendencias en las pruebas que se ejecutan con el resultado esperado.