INSTITUTO DEL SUR

DISEÑO DE SISTEMAS

Actividad de Aprendizaje

Desarrollo de Software

- Conceptos Preliminares
- Características

Conceptos: Diseño

- **Diseño:** se define como el proceso previo de configuración mental, en la búsqueda de una solución en cualquier campo. Se aplica habitualmente en el contexto de la industria, ingeniería, arquitectura, comunicación y otras disciplinas que requieren creatividad.

- **Diseño** se refiere a un **boceto**, **bosquejo** o **esquema** que se realiza, ya sea mentalmente o en un soporte material, antes de concretar la producción de algo. El término también se emplea para referirse a la **apariencia** de ciertos productos en cuanto a sus líneas, forma y funcionalidades.

Conceptos: Sistema

- **Sistema:** es un **conjunto de elementos relacionados entre sí** que funcionan como un todo, para lograr un objetivo común.

- Los elementos que componen un sistema pueden ser variados, como una serie de principios o reglas estructuradas sobre una materia o teoría.

Ejemplos de Sistemas

- Sistema solar.
- Sistema operativo.
- Sistema de información.
- Sistema educativo.

- Sistemas del cuerpo humano:
  - Sistema digestivo.
  - Sistema nervioso.

Conceptos: Diseño de Sistemas

- El **Diseño de sistemas:** es el arte de definir la arquitectura de hardware y software, componentes, módulos y datos de un sistema de cómputo, a efectos de satisfacer ciertos requerimientos. Es la etapa posterior al Análisis de sistemas y anterior al Desarrollo de sistemas.
- El **Diseño de sistemas:** tiene un rol respetado y crucial en la industria de procesamiento de datos. La importancia de los sistemas multiplataforma ha incrementado la ingeniería de software a costa de los diseños de sistemas.

Conceptos: Sistema de Información

- **Los Sistemas de Información:** ayudan a administrar, recolectar, recuperar, procesar, almacenar y distribuir información relevante para los procesos fundamentales y las particularidades de cada organización.

- La importancia de un **Sistema de Información** radica en la eficiencia en la correlación de una gran cantidad de datos ingresados a través de procesos diseñados para cada área con el objetivo de producir información válida para la posterior toma de decisiones.

Actividades básicas de un Sistema de Información

Actividades de un Sistema de Información

Componentes de un Sistema de Información

- **Entrada**: Proceso mediante el cual se captura y prepara datos para su posterior procesamiento. Las entradas pueden ser manuales o automáticas. Las manuales se realizan por el operador o el usuario, y las automáticas surgen de otros sistemas.
- **Almacenamiento:** Proceso mediante el cual el sistema almacena de manera organizada los datos e información para su uso posterior.
- **Proceso**: Es la capacidad de efectuar operaciones con los datos guardados en las unidades de memoria.
- **Salida**: Actividad que permite transmitir información útil y valiosa a los usuarios finales.

Componentes de un Sistema de Información

- Además un sistema de información debe tener control del desempeño del sistema, es decir debe generar **retroalimentación** sobre las actividades de entrada, procesamiento, almacenamiento y salida de información.
- Esta **retroalimentación** debe evaluarse para determinar si el sistema cumple con los estándares de desempeño establecidos.

Elementos de un Sistema de Información

Software:

- Son programas de comp utadora, con estructuras de datos y su documentación, que hacen efectiva la metodología de los requerimientos de los usuarios.
- El software da el soporte lógico y las ordenes al hardware.

Hardware:

- Son dispositivos electrónicos y electromecánicos, que proporcionan capacidad de cálculos y funciones rápidas, exactas y efectivas a las computadoras.
- El hardware da el soporte físico al software.

Elementos de un Sistema de Información

Gente:

- Los individuos que son usuarios y operadores del software y del hardware.

- **Documentación:** Los manuales, los impresos y otra información descriptiva que explica el uso y / o la operación.

Bases de Datos:

- Una colección grande y organizada de información a la que se accede mediante el software y que es una parte integral del funcionamiento del sistema.

Elementos de un Sistema de Información

Procedimientos:

- Son los pasos que definen el uso especifico de cada uno de los elementos o componentes del Sistema y las reglas de su manejo y mantenimiento.

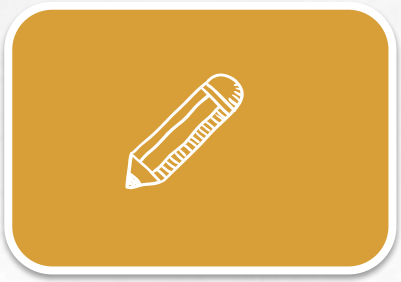
Control:

- Los sistemas trabajan mejor cuando operan dentro de los niveles de control tolerables de rendimiento.

Diseño de Sistemas de Información

Se define el proceso de "**aplicar**" ciertas técnicas y principios con el propósito de **"definir un dispositivo, un proceso o un Sistema",** con suficientes detalles como para permitir su interpretación y realización física.

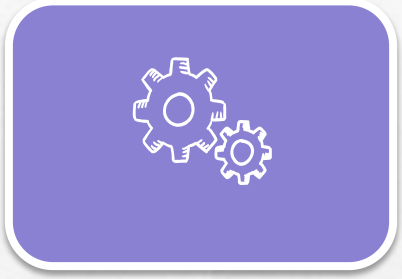
Etapas del Diseño de un Sistema de Información



Diseño de Datos

Diseño Arquitectónico

Diseño de la Interfaz



Diseño de Procedimientos

Etapas del Diseño de un Sistema de Información

Diseño de Datos:

- Trasforma el modelo de dominio de la información, creado durante el análisis, en las estructuras de datos necesarios para implementar el Software.

Etapas del Diseño de un Sistema de Información

Diseño Arquitectónico:

- Define la relación entre cada uno de los elementos estructurales del programa.

Etapas del Diseño de un Sistema de Información

Diseño de la Interfaz:

- Describe “como se comunica el Software consigo mismo”, con los sistemas que operan junto con el y con los operadores y usuarios que lo emplean.

Etapas del Diseño de un Sistema de Información

Diseño de Procedimientos:

- Transforma elementos estructurales de la arquitectura del programa.

La importancia del **Diseño del Sistemas** se puede definir en una sola palabra **Calidad**, dentro del diseño es donde se fomenta la calidad del Proyecto. El Diseño es la única manera de materializar con precisión los requerimientos del cliente.

Características del Diseño de Sistemas

- Eficacia.
- Facilidad de uso.
- Ahorro.
- Detección de Fallos.
- Clasificación de información.
- Identificación de necesidades del cliente

Criterios Técnicos para Evaluar un Diseño

- Un diseño debe presentar una organización jerárquica que haga un uso inteligente del control entre los componentes del software.
- El diseño debe ser modular, es decir, se debe hacer una partición lógica del Software.
- Un diseño debe contener abstracciones de datos y procedimientos.
- Debe conducir a interfaces que reduzcan la complejidad de las conexiones entre los módulos y el entorno exterior.

Importancia del Diseño de un Sistema

- Debe implementar todos los requisitos implícitos que desea el cliente.
- Debe ser una guía que puedan leer y entender los que construyan el código y los que prueban y mantienen el Software.
- Debe proporcionar una completa idea de lo que es el Software, enfocando los dominios de datos, funcional y comportamiento de la Implementación.

Herramientas para el Diseño de Sistemas

Herramientas para el Diseño de Sistemas

Herramientas de Especificación

Apoyan el proceso de formular las características que debe tener una aplicación, tales como entradas, salidas, procesamiento y especificaciones de control. Muchas incluyen herramientas para crear especificaciones de datos.

Herramientas para Presentación

Se utilizan para describir la posición de datos, mensajes y encabezados sobre las pantallas de las terminales, reportes y otros medios de entrada y salida.

Herramientas para el Diseño de Sistemas

Herramientas para el Desarrollo de Sistemas

Estas herramientas ayuda a los analistas a trasladar diseños en aplicaciones funcionales.

Herramientas para Ingeniería de Software

Apoyan el Proceso de formular diseños de Software, incluyendo procedimientos y controles, así como la documentación correspondiente.

Herramientas para el Diseño de Sistemas

Generadores de Código

Producen el código fuente y las aplicaciones a partir de especificaciones funcionales bien articuladas.

Herramientas para Pruebas

Apoyan la fase de la evaluación, incluyen facilidades para examinar la correcta operación del Sistema así como el grado de perfección alcanzado en comparación con las expectativas.

Decisiones que debe tomar un Diseñador de Sistemas

Organizar el sistema en subsistemas

Asignar los subsistemas a los procesadores y tareas

Seleccionar una aproximación para la administración de almacenes de datos

Manejar el acceso a recursos globales

Manejar las condiciones de contorno

Establecer las prioridades

Conclusiones

- En una organización o Empresa, el Análisis y Diseño de Sistemas, es el proceso de estudiar su Situación con la finalidad de observar como trabaja y decidir si es necesario realizar una mejora por medio de los resultados obtenidos.

- Mejora la calidad y eficiencia en el proceso de la toma de decisiones. Las decisiones podrán tomarse de una forma más ágil con el apoyo del análisis y diseño de un sistema de información.

30

Gracias por su atención!

“Dime y lo olvido, enséñame y lo recuerdo, involúcrame y lo aprendo”.

Benjamín Franklin

Metodologías para el Diseño de Sistemas

Introducción

- En la actualidad la mayoría de los usuarios de microcomputadoras tienen acceso a un sistema de información o forman parte del mismo.
- Todas las organizaciones cuentan con un sistema de información de algún tipo, que sus empleados deben utilizar.
- La creación o establecimiento de un nuevo sistema de información en la organización, puede ser una tarea compleja.
- Para encarar este tipo de situaciones existe un proceso de análisis y diseño de sistemas que auxilia en la resolución de tales problemas.

Introducción

- El análisis y diseño de sistemas proporciona una guía útil que busca disminuir las situaciones de fracaso o errores en estos procesos.

- Este procedimiento se lleva a cabo, en el llamado ciclo de vida de desarrollo de sistemas. Este ciclo puede repetirse indefinidamente, porque las organizaciones siempre se ven sometidas a cambios, y sus sistemas deben renovarse periódicamente.

Método y Metodología

- **Un método** es el procedimiento utilizado para llegar a un fin.
- Su significado original señala el camino que conduce a un lugar.
- **El método** es un orden que se debe imponer a los diferentes procesos necesarios para lograr un fin dado o resultados.
- En la ciencia se entiende por método al conjunto de procesos que el hombre debe emprender en la investigación y demostración de la verdad.

Método y Metodología

- **Metodología** es aquella guía que se sigue a fin realizar las acciones propias de una investigación. En términos más sencillos se trata de la guía que nos va indicando qué hacer y cómo actuar cuando se quiere obtener algún tipo de investigación.

- Es aplicable por ejemplo al ámbito laboral, donde tenemos una Metodología de Trabajo que nos lleva a lograr un mayor rendimiento y productividad, como también una Metodología de Estudio que nos permite alcanzar una mayor eficiencia a la hora de estudiar y realizar alguna labor educativa o didáctica.

Ciclo de Vida del Desarrollo de Software

Ciclo de Vida del Desarrollo de Software

- Es el proceso de dividir el trabajo de desarrollo del software en distintas fases para mejorar el diseño, la gestión del producto, y la gestión de proyecto.

- Es también conocido como el **ciclo de vida del desarrollo de software.**

Ciclo de Vida del Desarrollo de Software

Ciclo de Vida del Desarrollo de Software

- Cualquier sistema de información va pasando por una serie de fases a lo largo de su vida.
- Su ciclo de vida comprende una serie de etapas entre las que se encuentran las siguientes:

  - Planificación.
  - Análisis.
  - Diseño.
  - Implementación.
  - Pruebas.
  - Instalación o despliegue.
  - Uso y mantenimiento.

Ciclo de Vida del Desarrollo de Software

Planificación

- Comienza con un pedido escrito, que identifica el sistema de información y los cambios deseados. Pueden ser cambios mayores (un nuevo sistema) o cambios menores (un reporte).
- El propósito de la fase de planificación es identificar claramente la naturaleza y el alcance del problema.
- Se requiere una investigación preliminar y el resultado se llama Informe de Investigación Preliminar. La investigación preliminar también es conocida como **Estudio de Viabilidad**.

Ciclo de Vida del Desarrollo de Software

Planificación

- **Estudio de viabilidad**
  - Con recursos ilimitados (tiempo y dinero), casi cualquier proyecto se podría llevar a buen puerto. Por desgracia, en la vida real los recursos son más bien escasos, por lo que no todos los proyectos son viables.

Ciclo de Vida del Desarrollo de Software

Análisis

- En esta fase se recopilan y analizan los datos acerca del sistema y su funcionamiento aplicando cuestiones, entrevistas, encuestas, en general las técnicas de recopilación de datos.
- Especifica que es lo que el sistema debe hacer.

Ciclo de Vida del Desarrollo de Software

Diseño

- El propósito de esta fase es desarrollar un diseño (cómo va a quedar) del sistema de información que satisfaga todos los requisitos documentados.
- Se determina qué va a hacer el sistema. Se identifican las entradas, salidas, archivos, programas, procedimientos y controles del sistema.
- El documento creado se llama Especificaciones del Diseño del Sistema, debe ser aprobado por la gerencia y los usuarios.

Ciclo de Vida del Desarrollo de Software

Diseño

- Se ha de estudiar posibles alternativas de implementación para el sistema de información que vamos de construir y se ha de decidir la estructura general que tendrá el sistema (su diseño arquitectónico).
- El diseño de un sistema es complejo y el proceso de diseño ha de realizarse de forma iterativa.

Ciclo de Vida del Desarrollo de Software

Implementación

- Seleccionar las herramientas adecuadas, un entorno de desarrollo que facilite nuestro trabajo y un lenguaje de programación apropiado para el tipo de sistema que vayamos a construir.

- La elección de estas herramientas dependerá en gran parte de las decisiones de diseño que hayamos tomado hasta el momento y del entorno en el que nuestro sistema deberá funcionar.

Ciclo de Vida del Desarrollo de Software

Pruebas

- Tiene como objetivo detectar los errores que se hayan podido cometer en las etapas anteriores del proyecto (y, eventualmente, corregirlos).
- La búsqueda de errores que se realiza en la etapa de pruebas puede adaptar distintas formas, en función del contexto y de la fase del proyecto.

Ciclo de Vida del Desarrollo de Software

Instalación o despliegue

- Debemos de planificar el entorno en el que el sistema debe funcionar, tanto hardware como software: equipos necesarios y su configuración física, redes de interconexión entre los equipos y de acceso a sistemas externos, sistemas operativos y bibliotecas.

- Estas etapas son un reflejo del proceso que se sigue a la hora de resolver cualquier tipo de problema.

Ciclo de Vida del Desarrollo de Software

Uso y mantenimiento

La etapa de mantenimiento consume típicamente del 40 al 80 por ciento de los recursos de una empresa de desarrollo de software. De hecho, con un 60% de media, es probablemente la etapa más importante del ciclo de vida del software.

- Eliminar los defectos que se detecten durante su vida útil, lo primero que a uno se le viene a la cabeza cuando piensa en el mantenimiento de cualquier cosa.
- Adaptarlo a nuevas necesidades cuando el sistema ha de funcionar sobre una nueva versión del sistema operativo o en un entorno hardware diferente.
- Añadirle nueva funcionalidad, cuando se proponen características deseables que supondrían una mejora del sistema ya existente.

Proceso del desarrollo del software

- La mayoría de procesos de desarrollo modernos pueden ser vagamente descritos como ágiles.

- Otras metodologías incluyen:
  - Desarrollo en cascada.
  - Prototipado.
  - Desarrollo iterativo e incremental.
  - Desarrollo de espiral.
  - Desarrollo de aplicación rápida.
  - Programación extrema.

Desarrollo rápido de aplicaciones

- El desarrollo rápido de aplicaciones (RAD) es una metodología de desarrollo del software, que favorece desarrollo iterativo y la construcción rápida de prototipos en lugar de grandes cantidades de planificación inicial.

Desarrollo rápido de aplicaciones

Los principios básicos del desarrollo rápido de aplicaciones son:

- El objetivo clave es el rápido desarrollo y la entrega de un sistema de alta calidad a un costo de inversión relativamente bajo.
- Intenta reducir el riesgo inherente del proyecto dividiendo un proyecto en segmentos más pequeños y proporcionando más facilidad de cambio durante el proceso de desarrollo.
- La participación activa del usuario es imprescindible.
- Produce iterativamente software de producción, a diferencia de un prototipo desechable.
- Produce la documentación necesaria para facilitar el futuro desarrollo y mantenimiento.

Desarrollo en cascada

- Es un enfoque de desarrollo secuencial, en el que se considera que el desarrollo fluye constantemente hacia abajo (como una cascada) a través de varias fases, típicamente:
  - Análisis de requisitos que resulta en una especificación de requisitos de software.
  - Diseño del software
  - Implementación.
  - Testeo.
  - Integración, si hay múltiples subsistemas.
  - Despliegue (o Instalación).
  - Mantenimiento.

Desarrollo en cascada

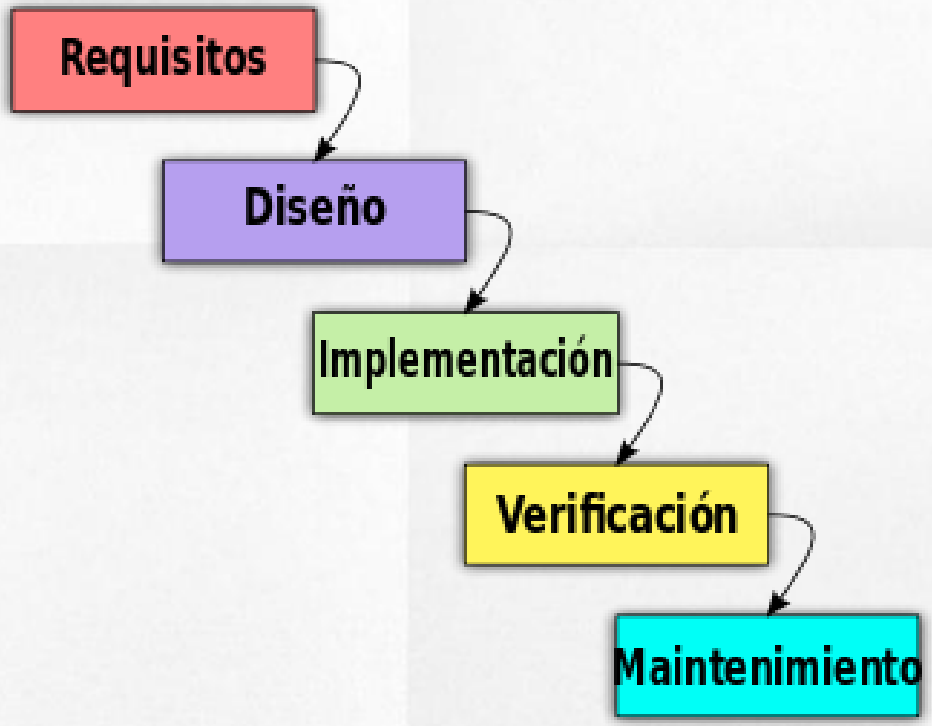
Los principios básicos son:

- El proyecto se divide en fases secuenciales, con algunas superposiciones y salpicaduras aceptables entre fases.

- El énfasis está en la planificación, los horarios, las fechas objetivo, los presupuestos y la implementación de todo un sistema a la vez.

- Se mantiene un estricto control a lo largo de la vida del proyecto a través de una extensa documentación escrita, revisiones formales y aprobación por parte del usuario, y la administración de la tecnología de la información que se realiza al final de la mayoría de las fases antes de comenzar la siguiente fase. La documentación escrita es un entregable explícito de cada fase.

Desarrollo en cascada



Prototipado

Consiste en la creación prototipos.

Los principios básicos son:

- El prototipado no es una metodología de desarrollo completa e independiente, sino más bien un enfoque para probar características particulares de otra metodología.
- Intentos de reducir los riesgos inherentes del proyecto a base de dividir un proyecto en pequeños segmentos proporcionando facilidad de cambio durante el proceso de desarrollo.
- El cliente está implicado durante el proceso de desarrollo, lo cual aumenta el probabilidad de que el cliente acepte la implementación final.

Prototipado

Consiste en la creación prototipos.

Los principios básicos son:

- Mientras algunos prototipos están desarrollados con la expectativa de que serán descartados, es posible que en algunos casos evolucionen de prototipo a sistema operativo.
- Una comprensión básica del problema fundamental del negocio es necesaria para evitar resolver los problemas equivocados, pero esto se cumple para todas las metodologías del software.

Desarrollo incremental

- Varios métodos son aceptables para combinar metodologías de desarrollo de sistemas lineales e iterativos, con el objetivo principal de reducir el riesgo inherente del proyecto al dividir un proyecto en segmentos más pequeños y proporcionar más facilidad de cambio durante el proceso de desarrollo.

Desarrollo incremental

Hay tres variantes principales de desarrollo incremental

- Se realiza una serie de mini cascadas, donde todas las fases de la cascada se completan para una pequeña parte de un sistema, antes de pasar al siguiente incremento.
- Los requisitos generales se definen antes de proceder al desarrollo evolutivo de mini cascadas de incrementos individuales de un sistema.
- El concepto inicial de software, el análisis de requisitos y el diseño de la arquitectura y el núcleo del sistema se definen a través del método cascada, seguido de una implementación incremental, que culmina con la instalación de la versión final, un sistema operativo.

Desarrollo de espiral

Los principios básicos son:

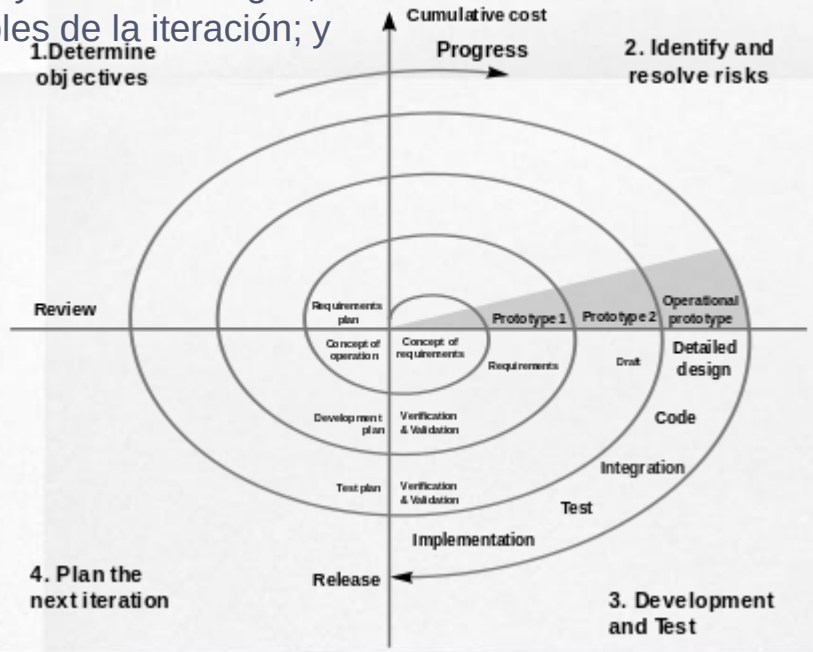
- Se centra la atención en la evaluación de riesgos y en minimizar el riesgo del proyecto al dividir un proyecto en segmentos más pequeños y brindar mayor facilidad de cambio durante el proceso de desarrollo, así como brindar la oportunidad de evaluar los riesgos y evaluar la continuación del proyecto a lo largo del ciclo de vida.

- "Cada ciclo implica una progresión a través de la misma secuencia de pasos, para cada parte del producto y para cada uno de sus niveles de elaboración, desde un documento de concepto de operación general hasta la codificación de cada programa individual".

Desarrollo de espiral

Los principios básicos son:

- Cada viaje alrededor de la espiral pasa por cuatro cuadrantes básicos:
  - (1) determinar objetivos, alternativas, y limitaciones de la iteración;
  - (2) evaluar alternativas; Identificar y resolver riesgos;
  - (3) desarrollar y verificar entregables de la iteración; y
  - (4) planear la iteración próxima.



Desarrollo de espiral

Los principios básicos son:

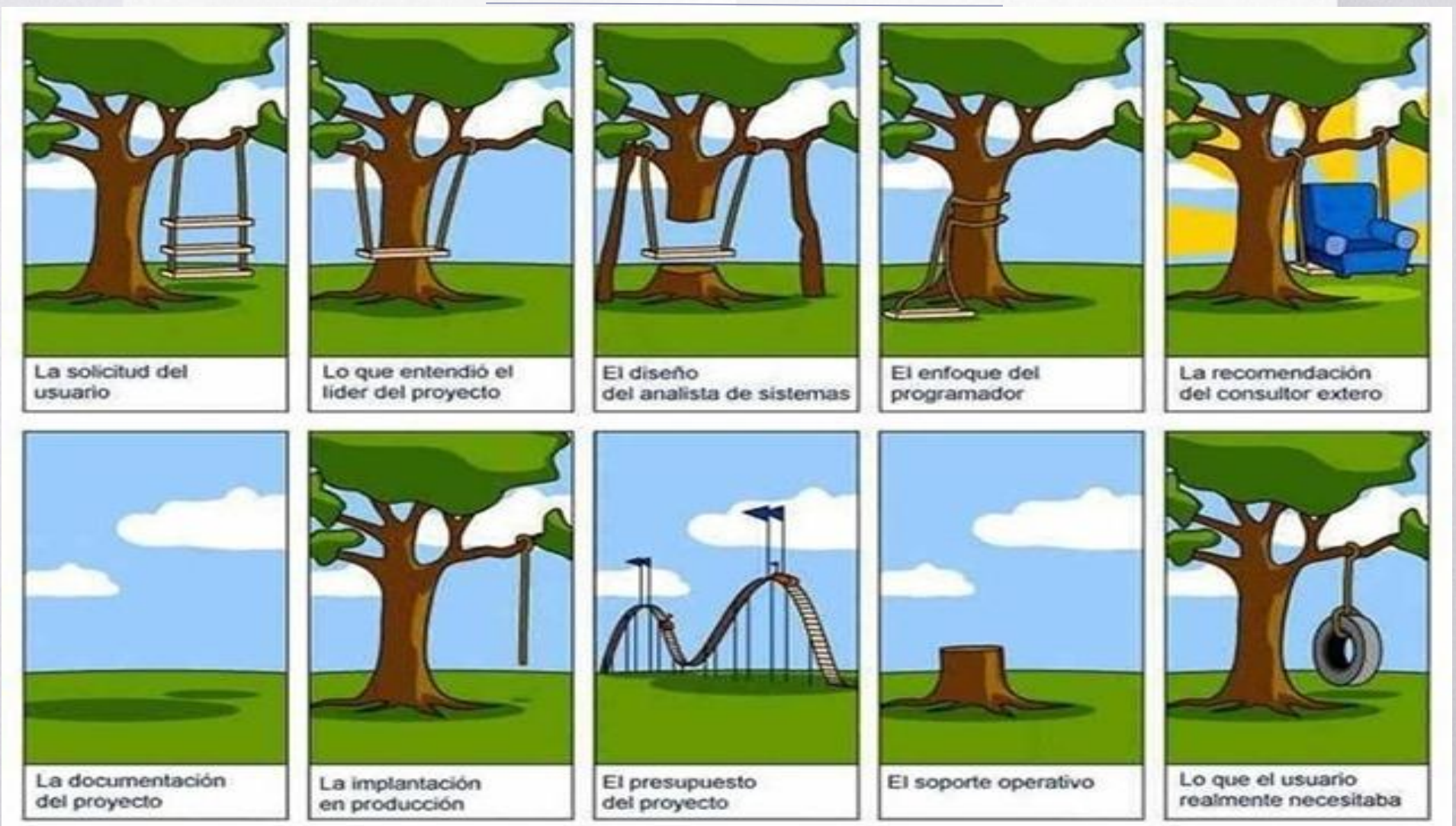
- Empieza cada ciclo con una identificación de las partes interesadas y sus "condiciones de victoria", y finaliza cada ciclo con una revisión y compromiso.

Programación extrema

- Es el más destacado de los procesos ágiles de desarrollo de software. Al igual que éstos, la programación extrema se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad.

- Los defensores de la XP consideran que los cambios de requisitos sobre la marcha son un aspecto natural, inevitable e incluso deseable del desarrollo de proyectos. Creen que ser capaz de adaptarse a los cambios de requisitos en cualquier punto de la vida del proyecto es una aproximación mejor y más realista que intentar definir todos los requisitos al comienzo del proyecto e invertir esfuerzos después en controlar los cambios en los requisitos.

CICLO DE VIDA DE UN SI



Proceso Unificado de Desarrollo

Proceso Unificado de Desarrollo Software

- Es un proceso de ingeniería de software que suministra un enfoque para asignar tareas y responsabilidades dentro de una organización de desarrollo.
- Su objetivo es asegurar la producción de software de alta y de mayor calidad para satisfacer las necesidades de los usuarios que tienen un cumplimiento al final dentro de un limite de tiempo y presupuesto previsible.
- Es una metodología de desarrollo iterativo que es enfocada hacia “ diagramas de los casos de uso, y manejo de los riesgos y el manejo de la arquitectura” como tal.

Proceso Unificado de Desarrollo Software

- El RUP mejora la productividad del equipo ya que permite que cada miembro del grupo sin importar su responsabilidad específica pueda acceder a la misma base de datos incluyendo sus conocimientos. Esto hace que todos compartan el mismo lenguaje, la misma visión y el mismo proceso acerca de cómo desarrollar un software.

- También permite evitar problemas legales ya que *Proceso Unificado de Rational* o *RUP* son marcas registradas por IBM (desde su compra de Rational Software Corporation en 2003).

Proceso Unificado de Desarrollo Software

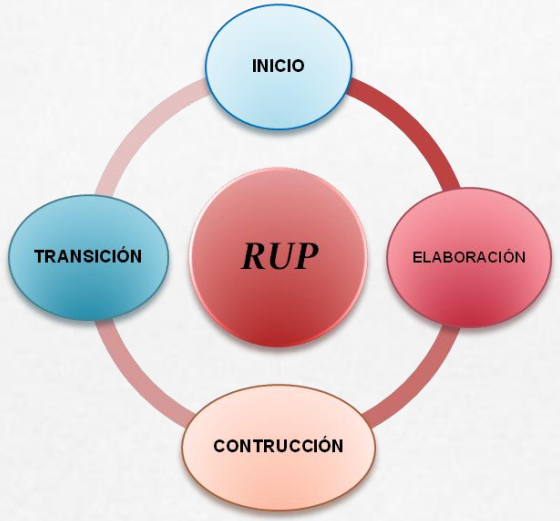
- Los autores que publican libros sobre el tema y que no están afiliados a Rational utilizan el término *Proceso Unificado*, mientras que los autores que pertenecen a Rational favorecen el nombre de *Proceso*

Proceso Unificado: Características principales

- **Iterativo e incremental.**
- **Dirigido por casos de uso.**
- **Centrado en la arquitectura.**
- **Enfocado en los riesgos**

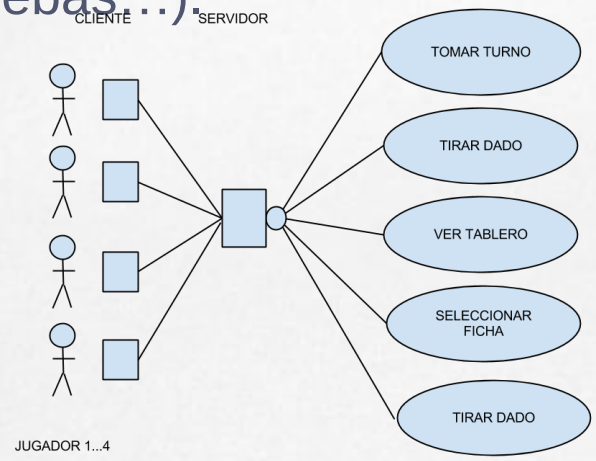
Proceso Unificado: Características principales

- **Iterativo e incremental**: cada iteración tiene 4 fases inicio, elaboración, construcción y transición. Estas iteraciones de estas fases, producen un incremento en el producto resultante añadiendo mejoras y nuevas funcionalidades.



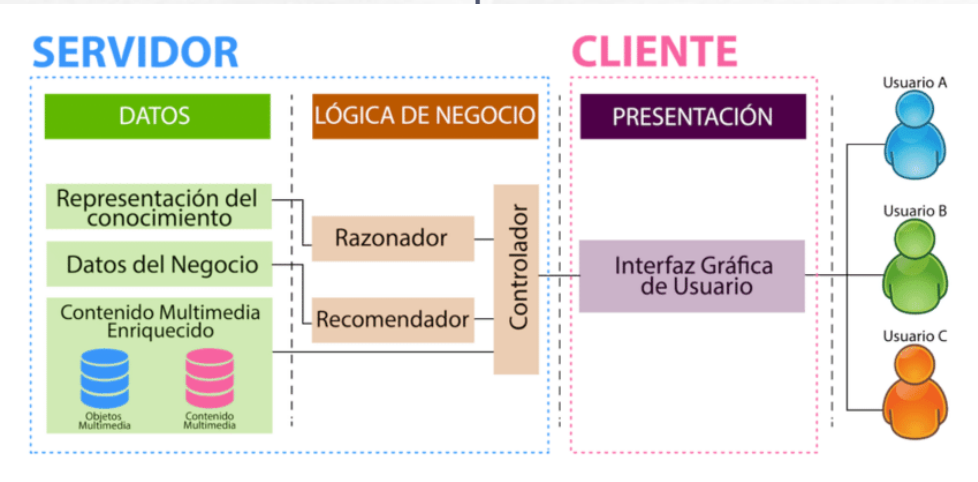
Proceso Unificado: Características principales

- **Dirigido por casos de uso**: Estos se utilizan para obtener los requisitos funcionales del sistema y así definir el contenido de cada una de las iteraciones. Así, la idea consiste en coger casos de uso o escenarios y desarrollar el proceso a través de las distintas disciplinas (diseño, implementación, pruebas…).



Proceso Unificado: Características principales

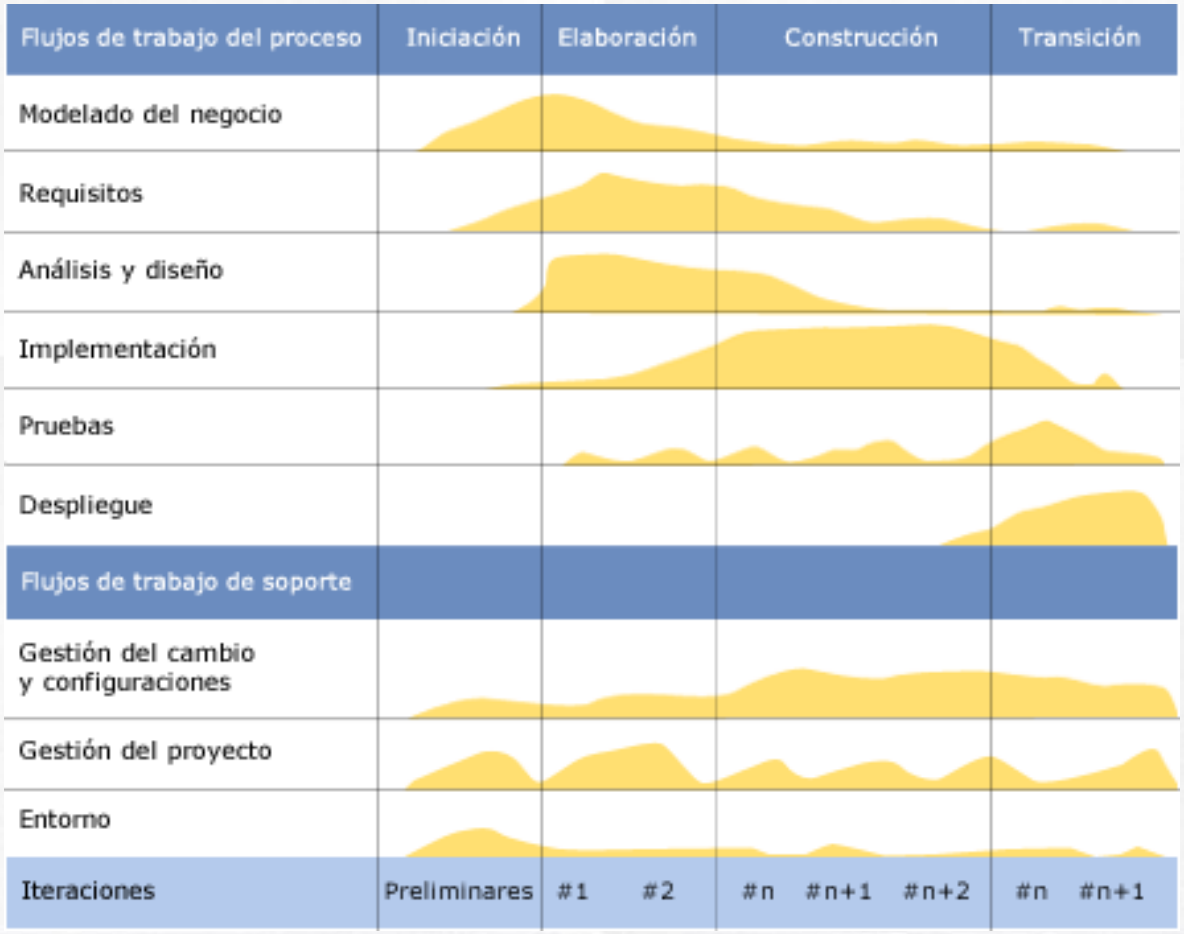
- **Centrado en la arquitectura**: Se asume que no existe un modelo único que cubra todos los aspectos del sistema, tal y como pasa con un edificio (hay diferentes planos para cada servicio), esta se debe respetar para que todo esté correctamente organizado lo que facilita el mantenimiento y sobre todo la facilidad de ampliación del sistema.



Proceso Unificado: Características principales

- **Enfocado en los riesgos:** El Proceso Unificado requiere que el equipo del proyecto se centre en identificar los riesgos críticos en una etapa temprana del ciclo de vida. Los resultados de cada iteración, en especial los de la fase de Elaboración deben ser seleccionados en un orden que asegure que los riesgos principales son considerados primero.

Ciclo de Vida de la Metodología RUP



Ciclo de Vida de la Metodología RUP

- **Fase de Inicio:** Esta fase tiene como propósito definir y acordar el alcance del proyecto con los patrocinadores, identificar los riesgos asociados al proyecto, proponer una visión muy general de la arquitectura de software y producir el plan de las fases y el de iteraciones posteriores.

- **Fase de elaboración:** En la fase de elaboración se seleccionan los casos de uso que permiten definir la arquitectura base del sistema y se desarrollaran en esta fase, se realiza la especificación de los casos de uso seleccionados y el primer análisis del dominio del problema, se diseña la solución preliminar.

Ciclo de Vida de la Metodología RUP

- **Fase de Construcción:** El propósito de esta fase es completar la funcionalidad del sistema, para ello se deben clarificar los requisitos pendientes, administrar los cambios de acuerdo a las evaluaciones realizadas por los usuarios y se realizan las mejoras para el proyecto.
- **Fase de Transición:** El propósito de esta fase es asegurar que el software esté disponible para los usuarios finales, ajustar los errores y defectos encontrados en las pruebas de aceptación, capacitar a los usuarios y proveer el soporte técnico necesario. Se debe verificar que el producto cumpla con las especificaciones entregadas por las personas involucradas en el proyecto.

Faces del Ciclo de Vida

- Establece oportunidad y alcance.
- Identifica las entidades externas o actores con las que se trata.
- Identifica los casos de uso.

RUP comprende 2 aspectos importantes por los cuales se establecen las disciplinas:

- **Proceso.**
- **Soporte.**

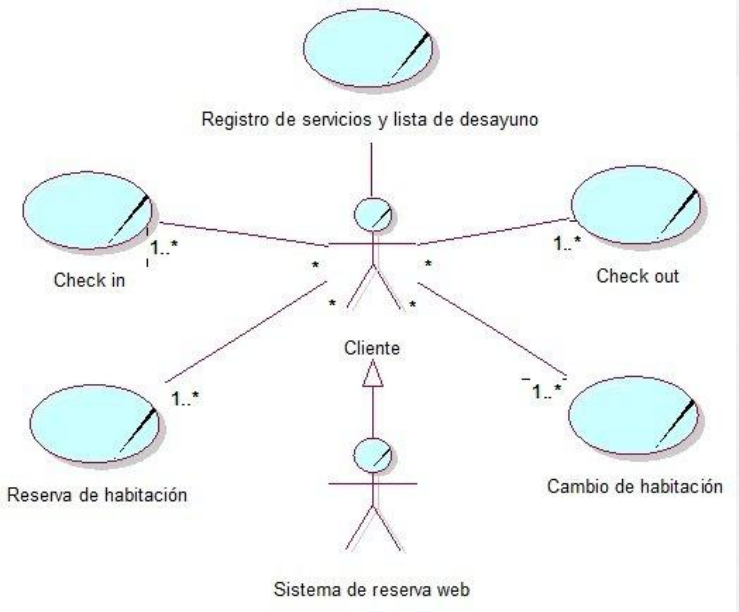
Proceso

Las etapas de esta sección son:

- Modelado de negocio.
- Requisitos.
- Análisis y Diseño.
- Implementación.
- Pruebas.
- Despliegue.

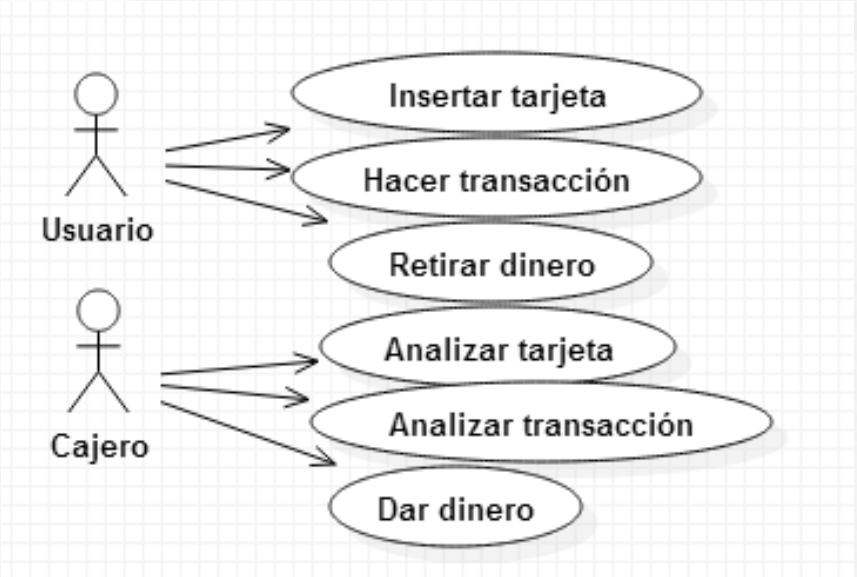
Proceso

- **1. Modelado de negocios:** Entiende los problemas e identifica mejoras potenciales, asegura que los participantes en este modelo tengan el entendimiento del problema, deriva los requerimientos del software.



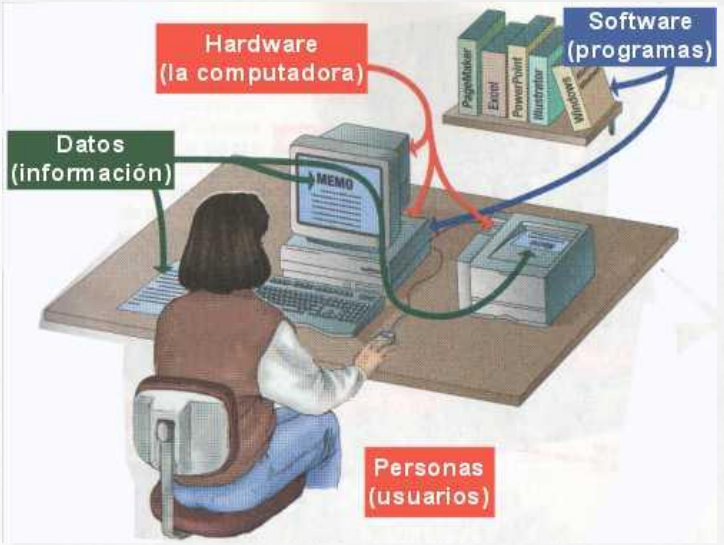
Proceso

- **2. Requerimientos:** Mantiene a los interesados sobre lo que el proyecto debe realizar, define los limites y requerimientos, se enfoca en las necesidades del usuario y hace una base de costos.



Proceso

- **3. Análisis y diseño:** Transforma los requerimientos al diseño y su arquitectura robusta y lo adapta para corresponder al ambiente de implementación y ajustarla para un desempeño esperado.



Proceso

- **4. Implementación:** Define el código, convierte el diseño en archivos ejecutables, prueba los componentes desarrollados como unidades, integra esas unidades en un sistema ejecutable.

Proceso

- **5. Pruebas:** Se enfoca en la evaluación de la calidad del producto, encuentra las fallas y las documenta, valida los requerimientos planteados y el buen funcionamiento.

Proceso

- **6. Despliegue:** Describe las actividades entre el aseguramiento de la entrega y disponibilidad del producto hacia el usuario final, hay un énfasis entre probar el software en el sitio de desarrollo.

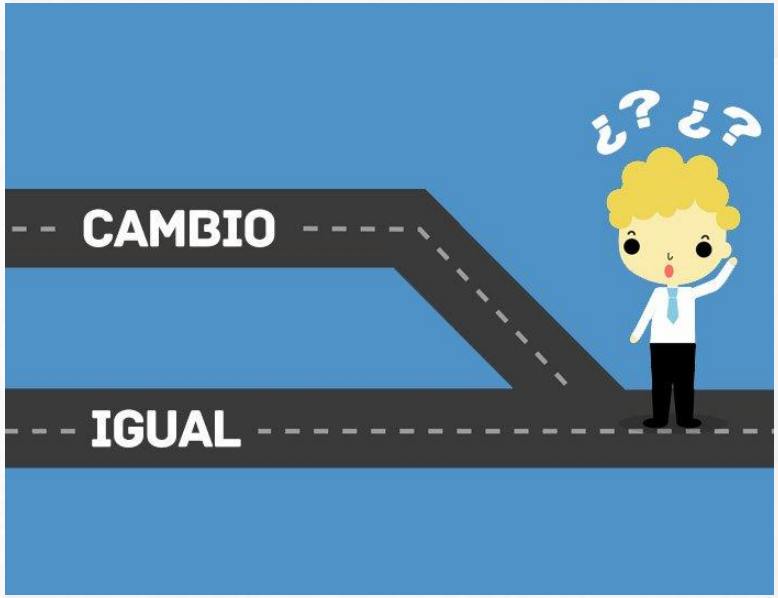
Soporte

En esta parte nos encontramos con las siguientes etapas:

- Gestión del cambio y configuraciones.
- Gestión del proyecto.
- Entorno.

Soporte

- **7. Gestión del cambio y configuraciones:** Consiste en controlar los cambios y mantiene la integridad de los productos que incluye el proyecto.



Soporte

- **8. Gestión del proyecto:** Provee un marco de trabajo para administrar los proyectos, guías para la planeación, soporte y ejecución, un marco de trabajo para administrar los riesgos.

Soporte

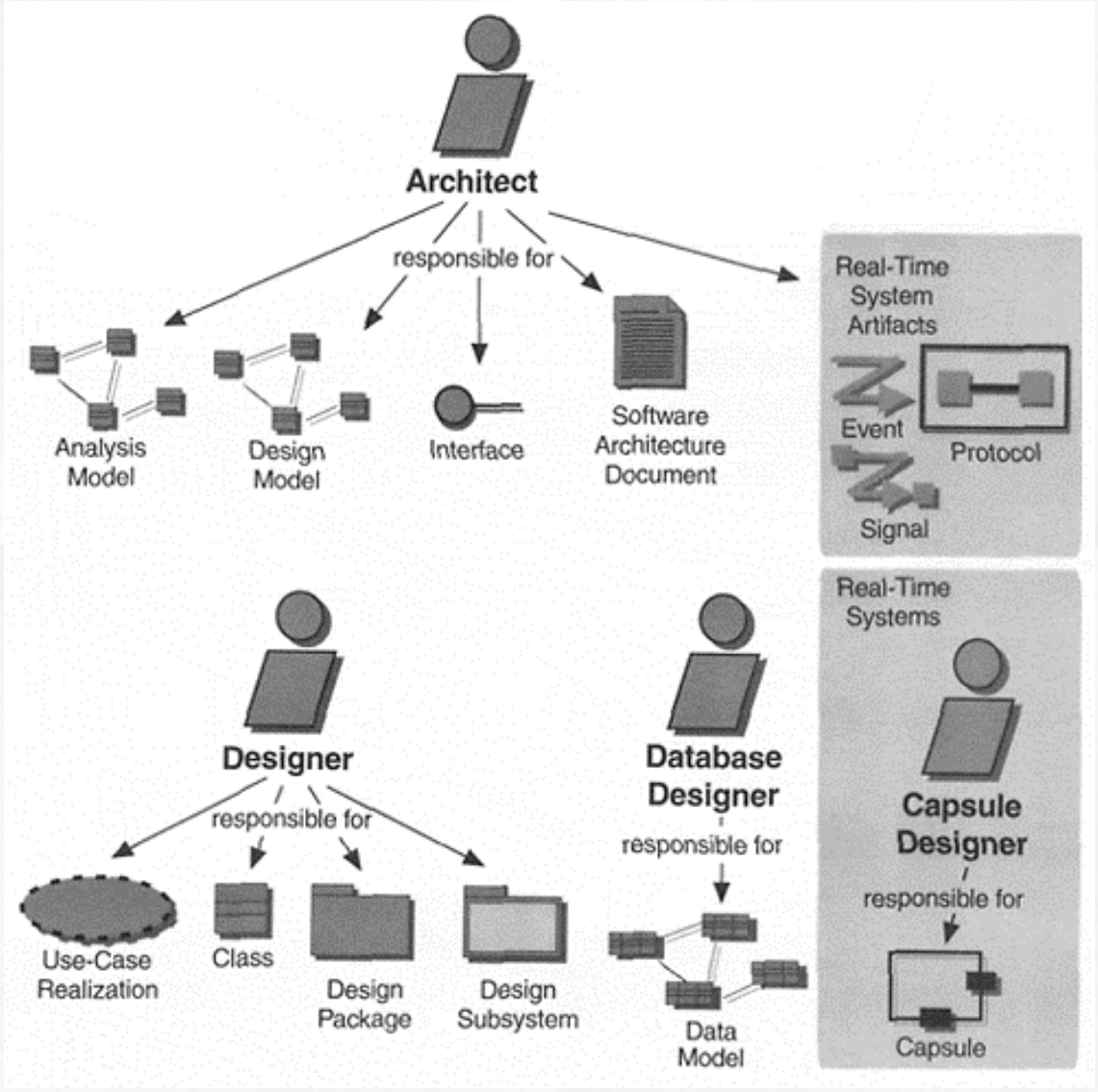
- **9. Entorno:** Se enfoca en las actividades para configurar el proceso del proyecto, describe las actividades requeridas para apoyar el proyecto, su propósito para proveer a las organizaciones de desarrollo de SW del ambiente necesario.

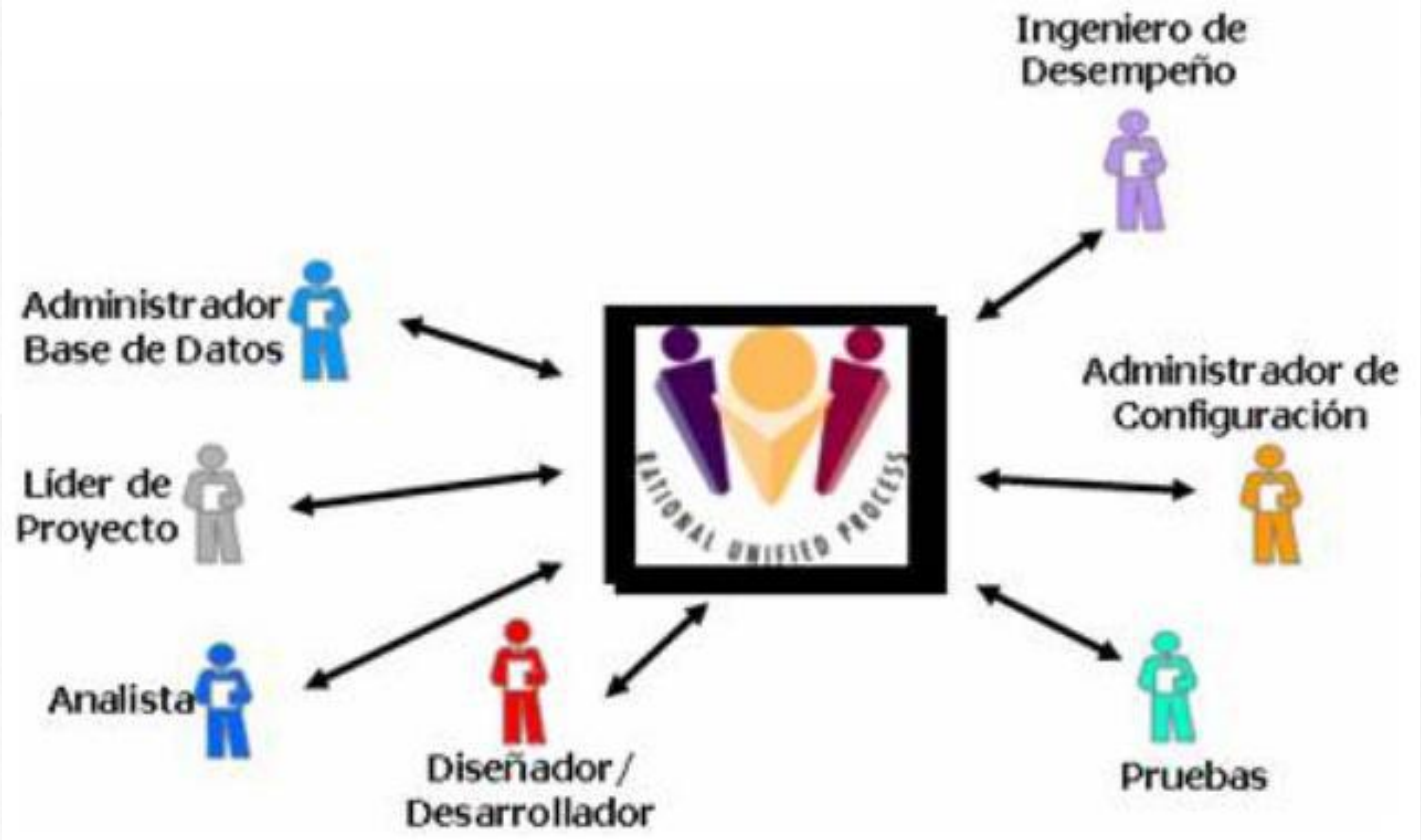
Análisis y Diseño de RUP

Análisis y Diseño de RUP

- En RUP se expresa el Análisis y proceso de Diseño en términos de:
  - Actividades,
  - Roles y
  - Artefactos,

como se muestra en la siguiente figura:


Architect
responsible for
Analysis Model
Design Model
Interface
Software Architecture Document
Real-Time System Artifacts
Event
Protocol
Signal
Designer
responsible for
Use-Case Realization
Class
Design Package
Design Subsystem
Database Designer
responsible for
Data Model
Real-Time Systems
Capsule Designer
responsible for
Capsule

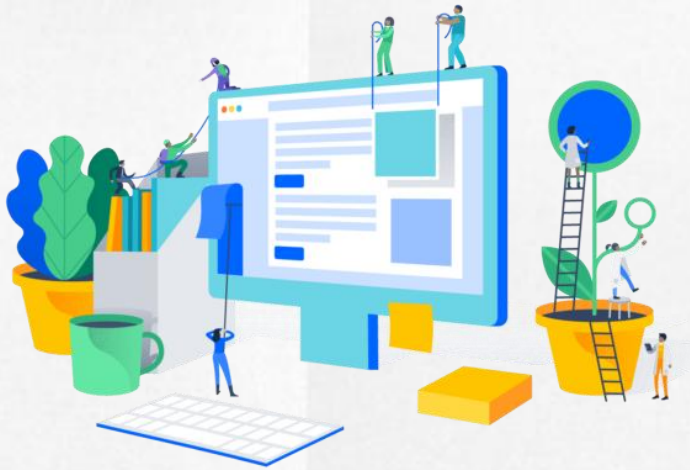

Ingeniero de Desempeño
Administrador Base de Datos
Administrador de Configuración
Líder de Proyecto
Analista
Diseñador / Desarrollador
Pruebas

Análisis y Diseño de RUP

- Los principales roles involucrados en el Análisis y el Diseño son:
  - **Arquitecto de software.**
  - **Diseñador.**

Arquitecto de software

- El arquitecto dirige y coordina las actividades técnicas y artefactos de todo el proyecto. Él o ella establece la estructura general de las interfaces entre las principales agrupaciones. En contraste con las opiniones de los demás roles, el arquitecto considera la amplitud en lugar de profundidad.



Arquitecto de software

- Se encarga de la definición de la arquitectura que guiará el desarrollo, y de la continua refinación de la misma en cada iteración; debe construir cualquier prototipo necesario para probar aspectos riesgosos desde el punto de vista técnico del proyecto; definirá los lineamientos generales del diseño y la implementación.

Diseñador

- El diseñador define las responsabilidades, las operaciones, atributos y relaciones de una o varias clases y determina la forma en que debe ajustarse al entorno de la implementación. Además, el diseñador puede tener la responsabilidad de uno o más paquetes de diseño o subsistemas de diseño, incluidas las clases de propiedad de los paquetes o subsistemas.

Diseñador

- Tiene a su cargo la codificación de los componentes en código fuente en algún lenguaje de programación durante cada iteración; debe elaborar y ejecutar las pruebas unitarias realizadas sobre el código desarrollado; es responsable de las clases que ha desarrollado debiendo documentarlas, actualizarlas ante cambios y mantenerlas bajo el control de configuración de las mismas mediante la herramienta utilizada.

Análisis y Diseño de RUP

- Existen roles opcionales que se pueden incluir en las actividades de Análisis y Diseño:

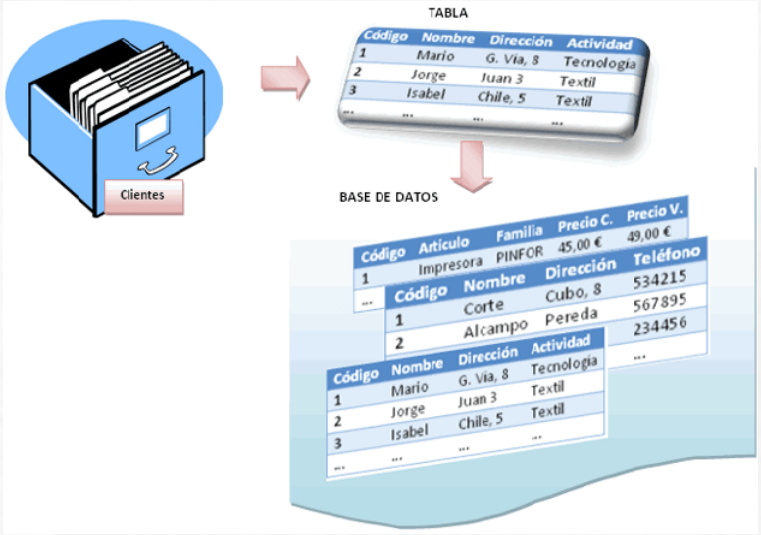
  - **Diseñador de base de datos.**
  - **Diseñador de Cápsulas. (para sistemas de tiempo real).**
  - **Revisor de Arquitectura o mentor.**
  - **Diseño Crítico o probador.**

Diseñador de base de datos

- El Diseñador de bases de datos es necesario cuando en el diseño del sistema se incluye una base de datos.

Diseñador de base de datos

- El diseñador de la base de datos define las tablas, índices, vistas, limitaciones, disparadores, procedimientos almacenados, tablas o parámetros de almacenamiento, y otras bases de datos específicas de las construcciones necesarias para almacenar, recuperar y eliminar la persistencia de objetos. Esta información se mantiene en el Artefacto: modelo de datos.

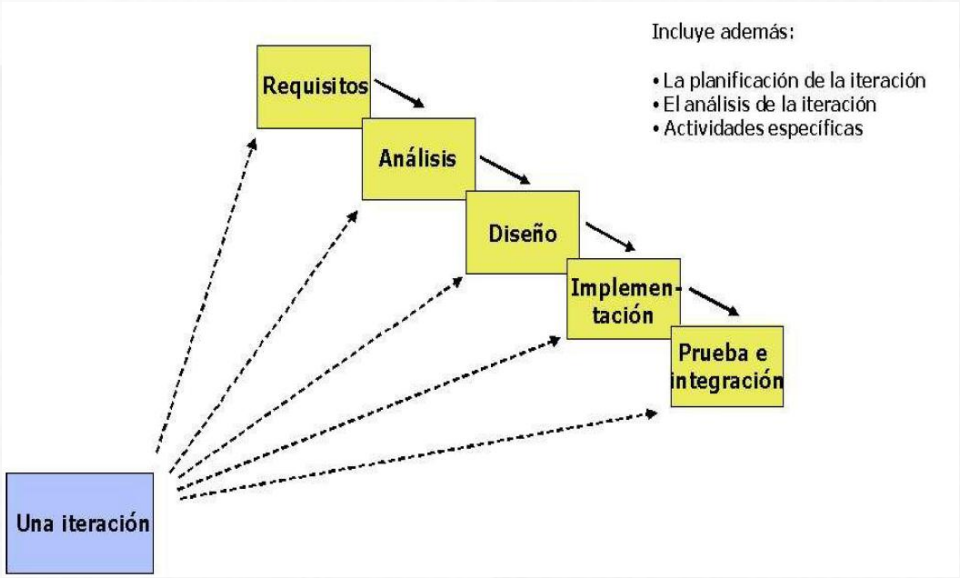


Diseñador de Cápsulas (para sistemas de tiempo real)

- El diseñador de la cápsula es un tipo de diseñador que se centra en garantizar que el sistema sea capaz de responder a los acontecimientos de manera oportuna, a través del uso adecuado de las técnicas de diseño.

Revisor de Arquitectura o mentor

- Es el rol que está íntimamente ligado con el proceso de desarrollo de software, que conoce todas las prácticas involucradas y entiende el porqué de la misma.



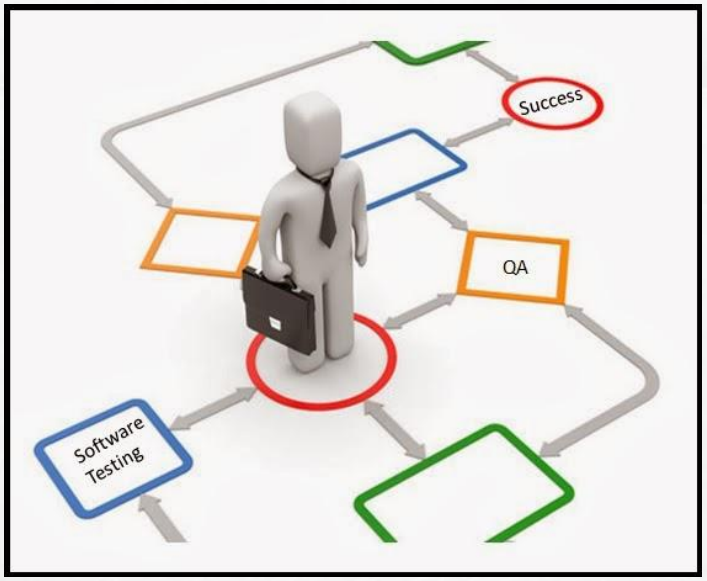
Revisor de Arquitectura o mentor

- Acompaña y apoya a los equipos de trabajo mediante revisiones de los artefactos y haciendo recomendaciones de cómo mejorar los mismos durante todo el ciclo de vida del sistema.

- Este rol está en capacidad de aclarar cualquier duda que puede surgir del proceso, así como también contribuye a que la calidad se mantenga durante el desarrollo del sistema.

Diseño Crítico o probador

- Estos especialistas revisan los principales artefactos producidos a través de este flujo de trabajo.

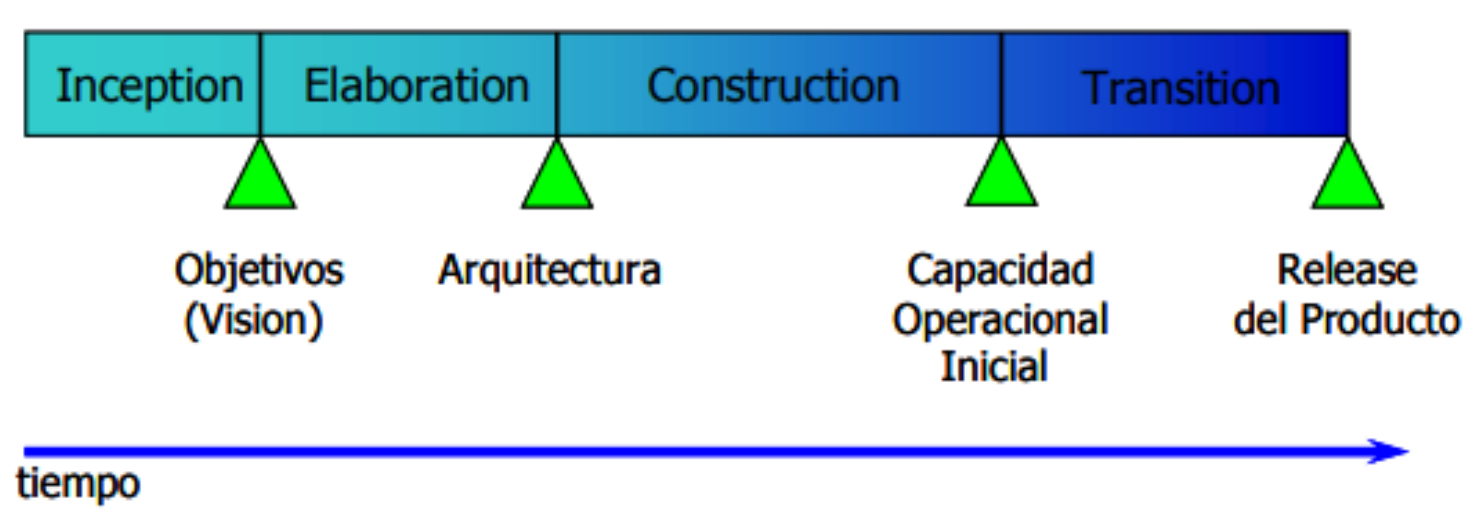


Artefactos

- Son los productos tangibles del proceso de desarrollo, como por ejemplo:
  - **Un documento**: como un Caso de Negocio o un documento de la arquitectura del Software.
  - **Un modelo**: como un modelo de caso de uso.
  - **Un elemento** de un modelo: como una sola clase de todo el Diagrama de Clases.

- RUP en cada una de sus fases realiza una serie de artefactos que sirven para comprender mejor tanto el análisis como el diseño del sistema (entre otros).

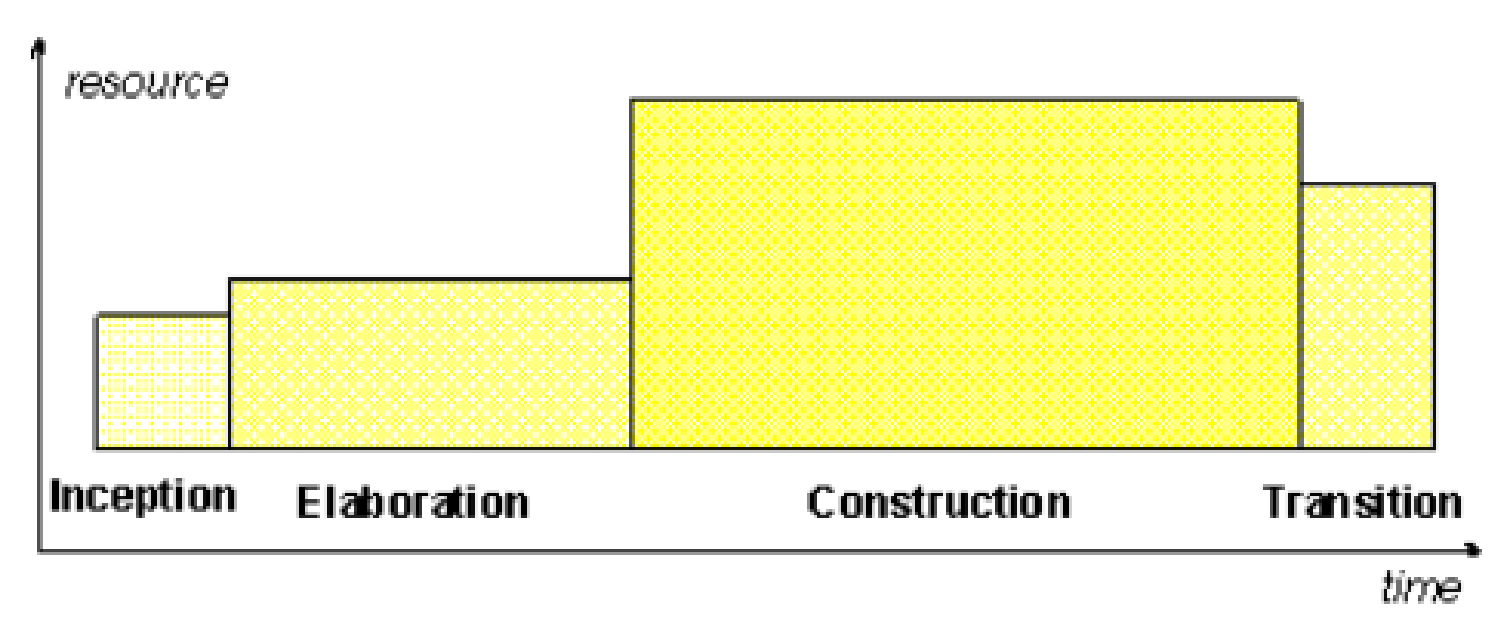
Fases e Hitos en RUP



Distribución del Esfuerzo y Tiempo

	Inicio	Elaboración	Construcción	Transición
Esfuerzo	5 %	20 %	65 %	10%
Tiempo Dedicado	10 %	30 %	50 %	10%

Distribución de Recursos Humanos



Inicio

- Durante la fase de inicio se define el modelo del negocio y el alcance del proyecto.
- Se identifican todos los actores y Casos de Uso, y se diseñan los Casos de Uso más esenciales (aproximadamente el 20% del modelo completo).
- Se desarrolla, un plan de negocio para determinar que recursos deben ser asignados al proyecto.

Inicio: Objetivos

Los objetivos de esta fase son:

- Establecer el ámbito del proyecto y sus límites.
- Encontrar los Casos de Uso críticos del sistema, los escenarios básicos que definen la funcionalidad.
- Mostrar al menos una arquitectura candidata para los escenarios principales.
- Estimar el coste en recursos y tiempo de todo el proyecto.
- Estimar los riesgos, las fuentes de incertidumbre.

Inicio: Artefactos

Los artefactos de la fase de inicio deben ser:

- Un documento de visión: Una visión general de los requerimientos del proyecto, características clave y restricciones principales.
- Modelo inicial de Casos de Uso (10-20% completado).
- Un glosario inicial: Terminología clave del dominio.
- El caso de negocio.
- Lista de riesgos y plan de contingencia.
- Plan del proyecto, mostrando fases e iteraciones.
- Modelo de negocio, si es necesario
- Prototipos exploratorios para probar conceptos o la arquitectura candidata.

Inicio: Hito

Al terminar la fase de inicio se deben comprobar los criterios de evaluación para continuar:

- Todos los interesados en el proyecto coinciden en la definición del ámbito del sistema y las estimaciones de agenda.
- Entendimiento de los requisitos, como evidencia de la fidelidad de los Casos de Uso principales.
- Las estimaciones de tiempo, coste y riesgo son creíbles.
- Comprensión total de cualquier prototipo de la arquitectura desarrollado.
- Los gastos hasta el momento se asemejan a los planeados.

Inicio: Hito

Si el proyecto no pasa estos criterios hay que plantearse abandonarlo o repensarlo profundamente.

Elaboración

- El propósito de la fase de elaboración es analizar el dominio del problema, establecer los cimientos de la arquitectura, desarrollar el plan del proyecto y eliminar los mayores riesgos. En esta fase se construye un prototipo de la arquitectura, que debe evolucionar en iteraciones sucesivas hasta convertirse en el sistema final. Este prototipo debe contener los Casos de Uso críticos identificados en la fase de inicio. También debe demostrarse que se han evitado los riesgos más graves.

Elaboración: Objetivos

Los objetivos de esta fase son:

- Definir, validar y cimentar la arquitectura.
- Completar la visión.
- Crear un plan fiable para la fase de construcción. Este plan puede evolucionar en sucesivas iteraciones. Debe incluir los costes si procede.
- Demostrar que la arquitectura propuesta soportará la visión con un coste razonable y en un tiempo razonable.

Elaboración: Artefactos

Al terminar deben obtenerse los siguientes resultados :

- Un modelo de Casos de Uso completa al menos hasta el 80%: todos los casos y actores identificados, la mayoría de los casos desarrollados.
- Requisitos adicionales que capturan los requisitos no funcionales y cualquier requisito no asociado con un Caso de Uso específico.
- Descripción de la arquitectura software.
- Un prototipo ejecutable de la arquitectura.

Elaboración: Artefactos

- Lista de riesgos y caso de negocio revisados.
- Plan de desarrollo para el proyecto.
- Un caso de desarrollo actualizado que especifica el proceso a seguir.
- Un manual de usuario preliminar (opcional).

En esta fase se debe tratar de abarcar todo el proyecto con la profundidad mínima. Sólo se profundiza en los puntos críticos de la arquitectura o riesgos importantes. En la fase de elaboración se actualizan todos los productos de la fase de inicio.

Elaboración: Hitos

Los criterios de evaluación de esta fase son los siguientes:

- La visión del producto es estable.
- La arquitectura es estable.
- Se ha demostrado mediante la ejecución del prototipo que los principales elementos de riesgo han sido abordados y resueltos.
- El plan para la fase de construcción es detallado y preciso. Las estimaciones son creíbles.
- Todos los interesados coinciden en que la visión actual será alcanzada si se siguen los planes actuales en el contexto de la arquitectura actual.
- Los gastos hasta ahora son aceptables, comparados con los previstos.

Si no se superan los criterios de evaluación quizá sea necesario abandonar el proyecto o replanteárselo considerablemente

Elaboración: Hito

Elaboración: Vistas

- Documento Arquitectura que trabaja con las siguientes vistas:

Vista Lógica

- Diagrama de clases
- Modelo E-R (Si el sistema así lo requiere)

Vista de Implementación

- Diagrama de Secuencia
- Diagrama de estados
- Diagrama de Colaboración

Vista Conceptual

- Modelo de dominio

Vista física

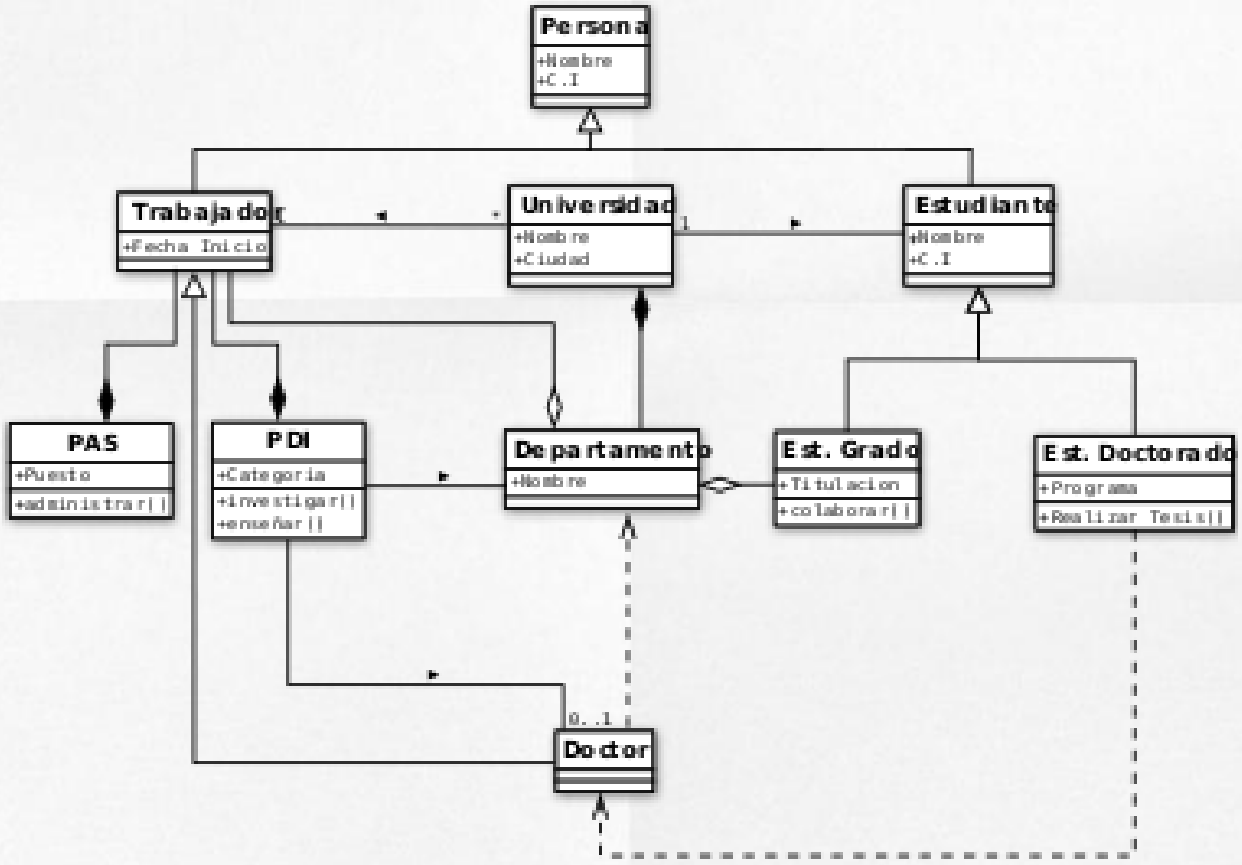
- Mapa de comportamiento a nivel de hardware.

Vista Lógica: Diagrama de clases

- Un **diagrama de clases** es una herramienta para comunicar el diseño de un programa que se creó y que permite modelar relaciones entre diferentes entidades.

- Es un tipo de **diagrama** que describe la estructura de un sistema mostrando las **clases** del sistema, sus atributos, operaciones (o métodos), y las relaciones entre los objetos.

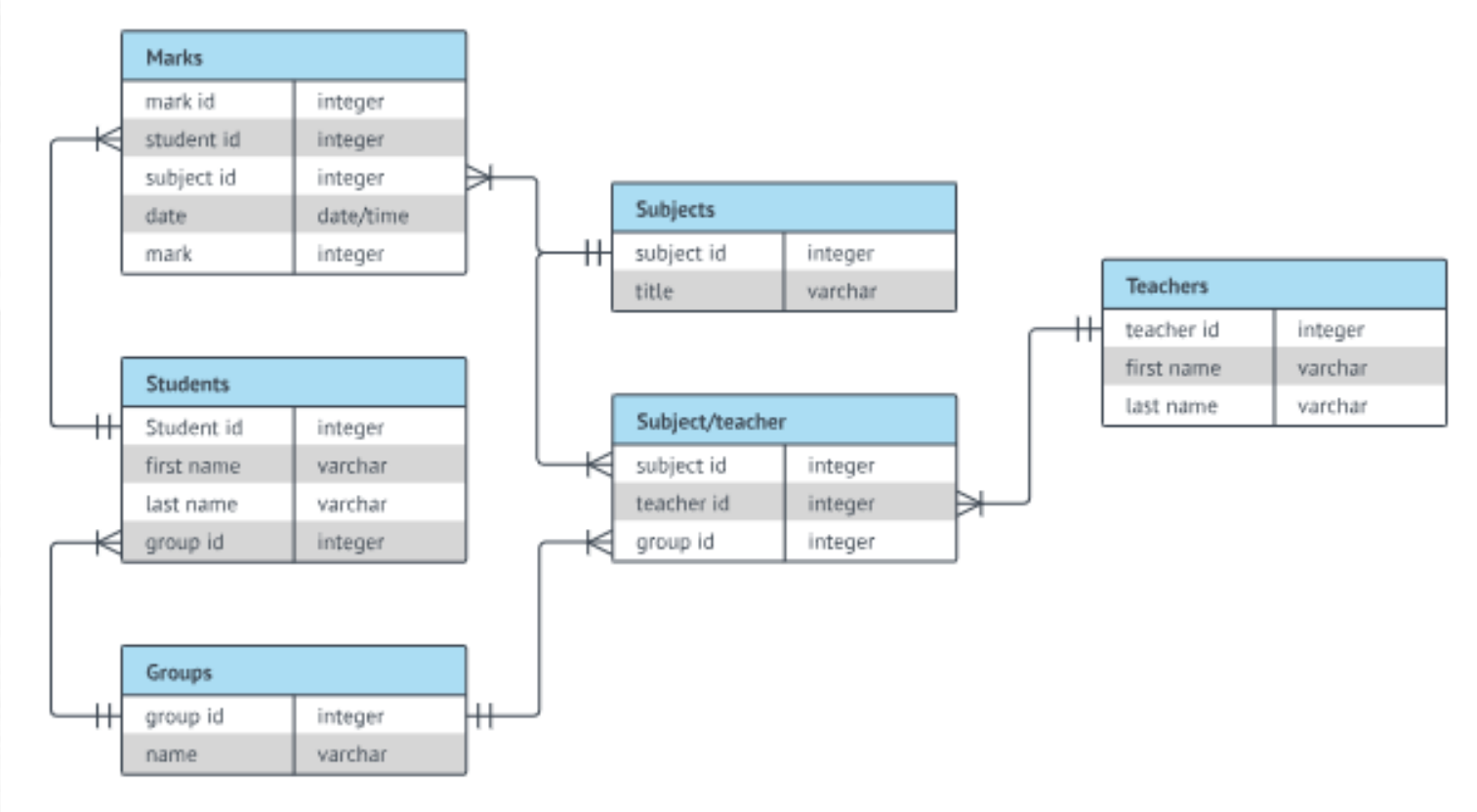
Vista Lógica: Diagrama de clases



Vista Lógica: Modelo E-R

- Un diagrama entidad-relación, también conocido como modelo entidad relación o ERD, es un tipo de diagrama de flujo que ilustra cómo las "entidades", personas, objetos o conceptos, se relacionan entre sí dentro de un sistema.

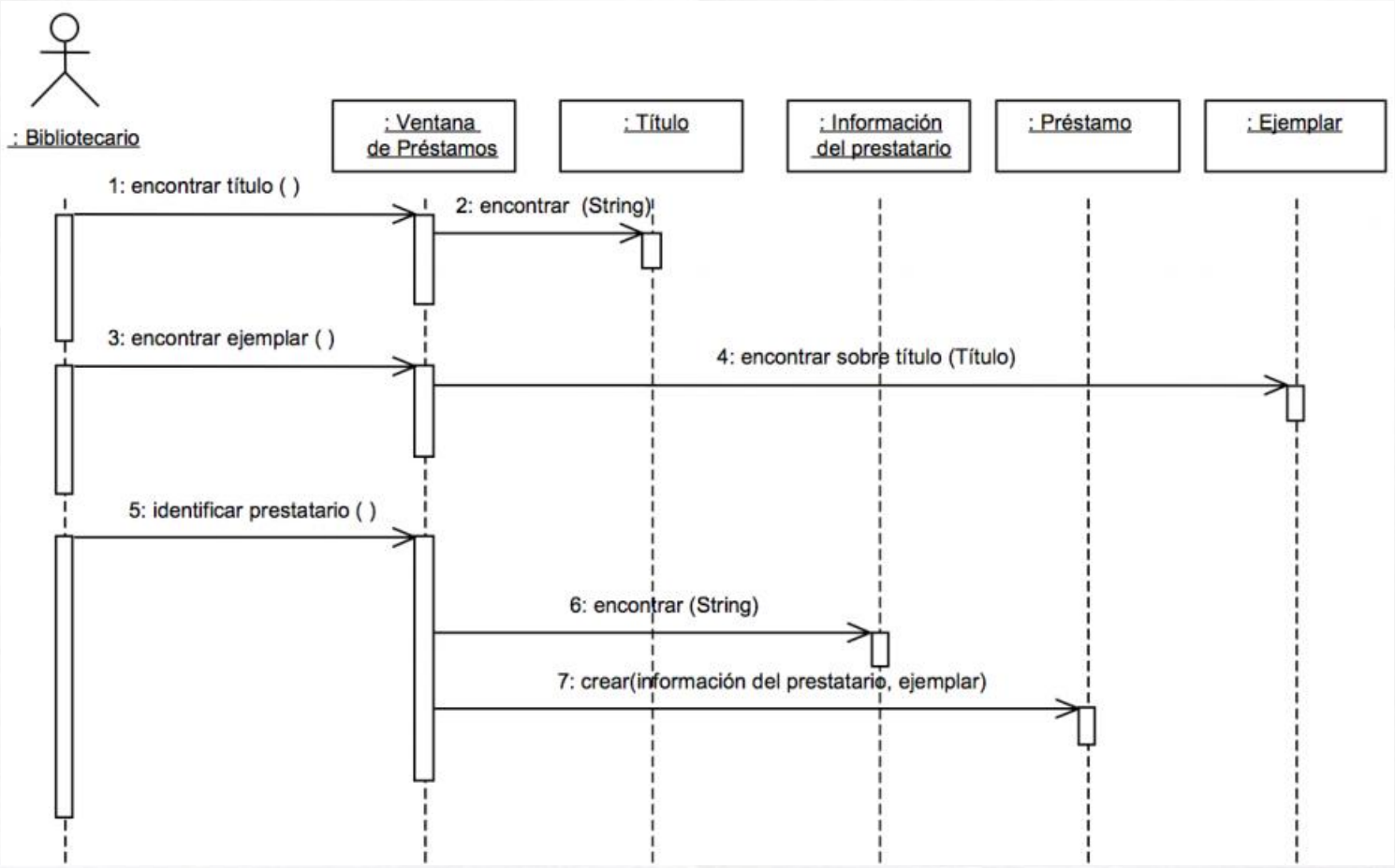
Vista Lógica: Modelo E-R



Vista de Implementación:
Diagrama de Secuencia

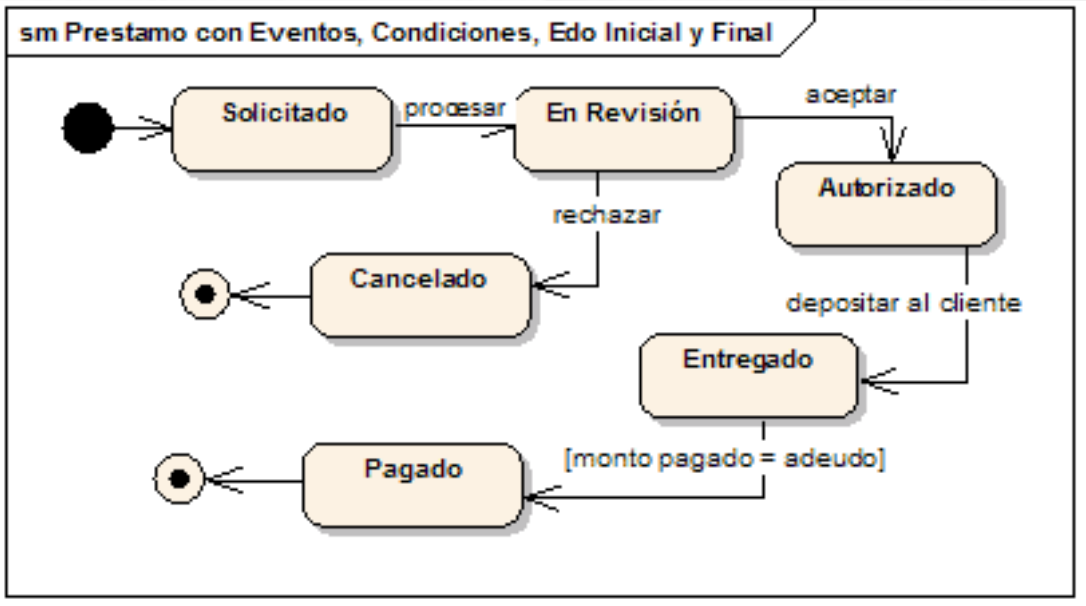
- El diagrama de secuencia es un tipo de diagrama de interacción cuyo objetivo es describir el comportamiento dinámico del sistema de información haciendo énfasis en la secuencia de los mensajes intercambiados por los objetos.

Vista de Implementación: Diagrama de Secuencia



Vista de Implementación: Diagrama de estados

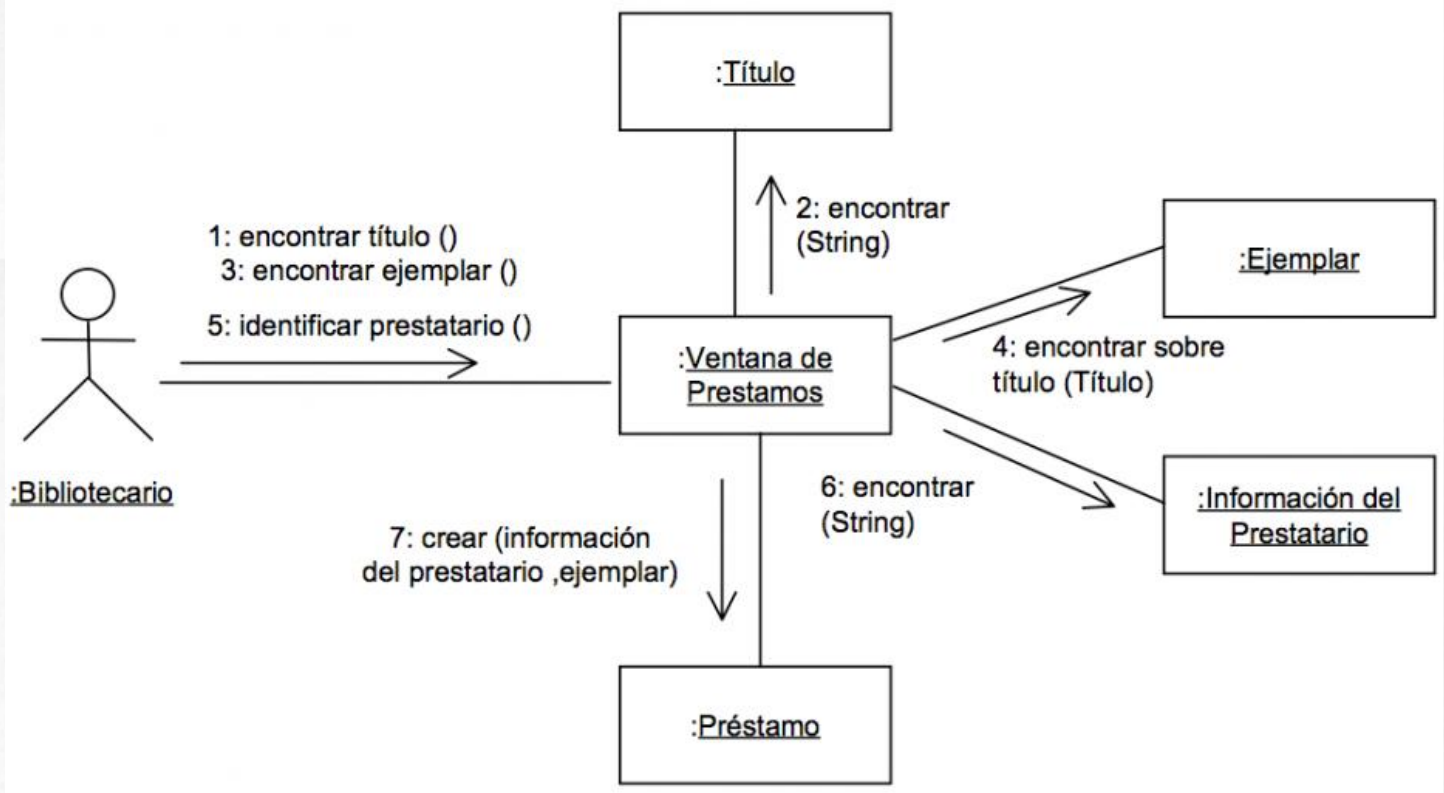
- Este diagrama muestra la secuencia de estados por los que pasa un caso de uso, un objeto a lo largo de su vida, o bien todo el sistema.



Vista de Implementación: Diagrama de Colaboración

- El diagrama de colaboración es un tipo de diagrama de interacción cuyo objetivo es describir el comportamiento dinámico del sistema de información mostrando como interactúan los objetos entre sí, es decir, con qué otros objetos tiene vínculos o intercambia mensajes un determinado objeto.

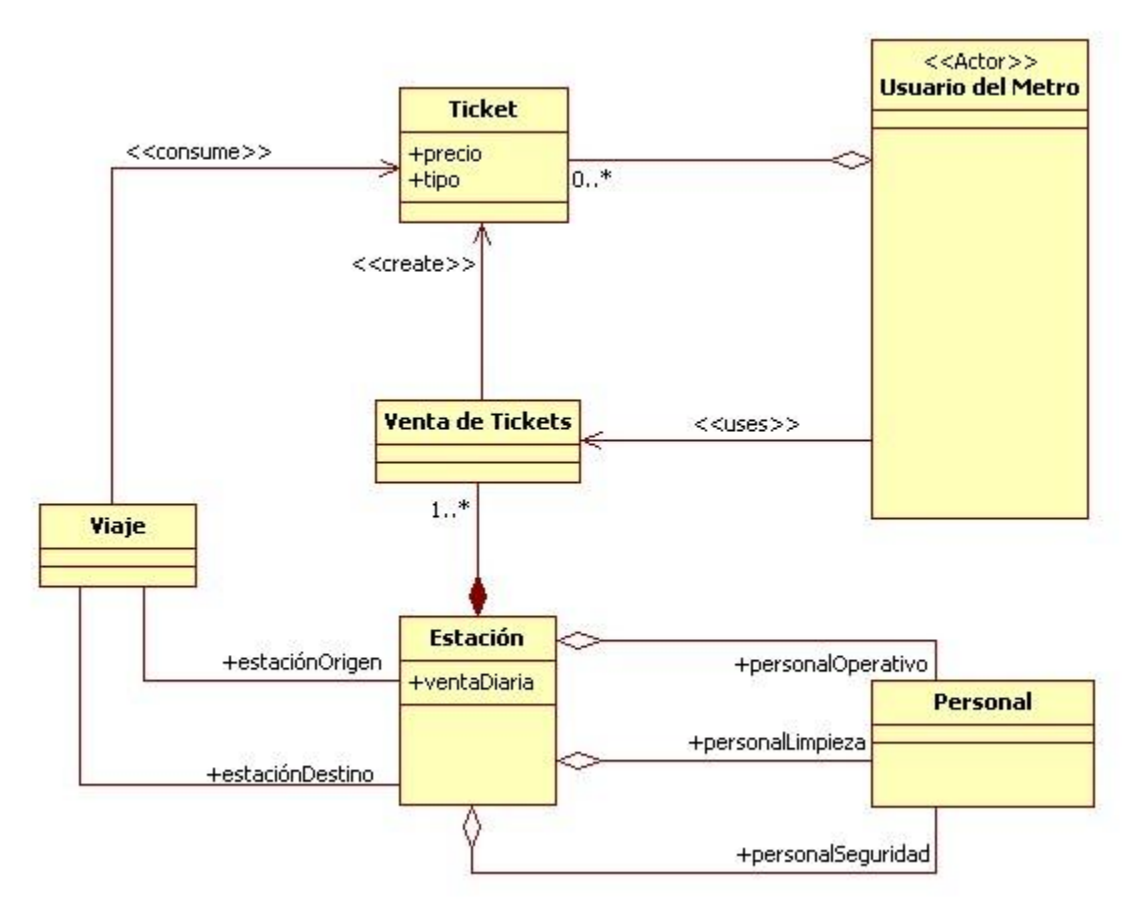
Vista de Implementación: Diagrama de Colaboración



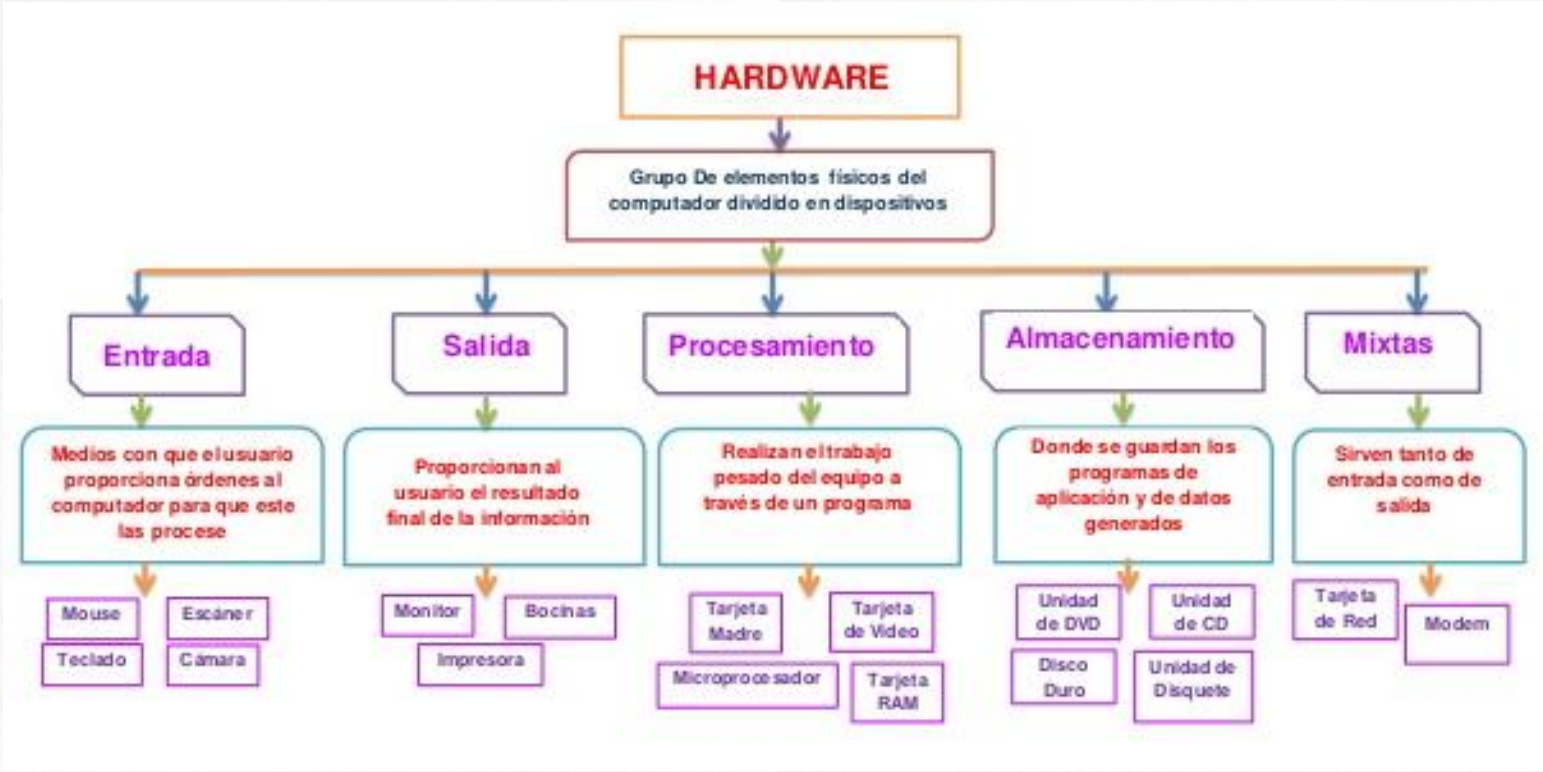
Vista Conceptual: Modelo de dominio

- El modelo de dominio puede ser tomado como el punto de partida para el diseño del sistema.
- Cuando se realiza la programación orientada a objetos, el funcionamiento interno del software va a imitar en alguna medida a la realidad, por lo que el mapa de conceptos del modelo de dominio constituye una primera versión del sistema.

Vista Conceptual: Modelo de dominio



Vista física: Mapa de comportamiento a nivel de hardware



Construcción

- La finalidad principal de esta fase es alcanzar la capacidad operacional del producto de forma incremental a través de las sucesivas iteraciones.
- Durante esta fase todos los componentes, características y requisitos deben ser implementados, integrados y probados en su totalidad, obteniendo una versión aceptable del producto.

Construcción: Objetivos

Los objetivos concretos incluyen:

- Minimizar los costes de desarrollo mediante la optimización de recursos y evitando el tener que rehacer un trabajo o incluso desecharlo.
- Conseguir una calidad adecuada tan rápido como sea práctico.
- Conseguir versiones funcionales (alfa, beta, y otras versiones de prueba) tan rápido como sea práctico.

Construcción: Artefactos

Los resultados de la fase de construcción deben ser:

- Modelos Completos (Casos de Uso, Análisis, Diseño, Despliegue e Implementación)
- Arquitectura íntegra (mantenida y mínimamente actualizada)
- Riesgos Presentados Mitigados
- Plan del Proyecto para la fase de Transición.
- Manual Inicial de Usuario (con suficiente detalle)
- Prototipo Operacional – beta
- Caso del Negocio Actualizado

Construcción: Hitos

Los criterios de evaluación de esta fase son los siguientes:

- El producto es estable y maduro como para ser entregado a la comunidad de usuario para ser probado.
- Todos los usuarios expertos están listos para la transición en la comunidad de usuarios.
- Son aceptables los gastos actuales versus los gastos planeados.

Construcción: Hito

Hito:

Transición

La finalidad de la fase de transición es poner el producto en manos de los usuarios finales, para lo que se requiere desarrollar nuevas versiones actualizadas del producto, completar la documentación, entrenar al usuario en el manejo del producto, y en general tareas relacionadas con el ajuste, configuración, instalación y facilidad de uso del producto.

Transición

Algunas de las cosas que puede incluir esta fase:

- Prueba de la versión Beta para validar el nuevo sistema frente a las expectativas de los usuarios
- Funcionamiento paralelo con los sistemas legados que están siendo sustituidos por nuestro proyecto.
- Conversión de las bases de datos operacionales.
- Entrenamiento de los usuarios y técnicos de mantenimiento.
- Traspaso del producto a los equipos de marketing, distribución y venta.

Transición: Objetivos

Los principales objetivos de esta fase son:

- • Conseguir que el usuario se valga por si mismo.
- • Un producto final que cumpla los requisitos esperados, que funcione y satisfaga suficientemente al usuario.

Transición: Artefactos

Los resultados de la fase de transición son :

- Prototipo Operacional
- Documentos Legales
- Caso del Negocio Completo
- Línea de Base del Producto completa y corregida que incluye todos los modelos del sistema
- Descripción de la Arquitectura completa y corregida
- Las iteraciones de esta fase irán dirigidas normalmente a conseguir una nueva versión.

Transición: Hitos

Los criterios de evaluación de esta fase son los siguientes:

- El usuario se encuentra satisfecho.
- Son aceptables los gastos actuales versus los gastos planificados.

Transición: Hito

Estado de Aspectos de los Casos de Usos al finalizar cada Fase

	Modelo de Negocio Terminado	Casos de Uso Identificados	Casos de Uso Descritos	Casos de Uso Analizados	Casos de Uso Diseñados, Implementados y Probados
Fase de Concepción	50% - 70%	50%	10%	5%	Muy poco, puede que sólo algo relativo a un prototipo para probar conceptos
Fase de Elaboración	Casi el 100%	80% o más	40% - 80%	20% - 40%	Menos del 10%
Fase de Construcción	100%	100%	100%	100%	100%
Fase de Transición					

Flujos de trabajo de RUP (Actividades)

Actividades

- Una actividad es una unidad de trabajo que se asigna a un trabajador. Ej.:
  - Crear o modificar un artefacto
- Una actividad lleva entre un par de horas y un par de días, involucra un solo trabajador y un número pequeño de artefactos.
- Las actividades se consideran en la planificación y evaluación del progreso del proyecto.
- Ejemplos:
  - Planificar una iteración - Administrador de proyecto
  - Encontrar actores y casos de uso - Analista
  - Revisar el diseño - Revisor de diseño
  - Ejecutar pruebas de performance - Ing. de pruebas de performance

Asignación de Actividades

Flujos de trabajo de RUP (Actividades)

- En RUP se han agrupado las actividades en grupos lógicos definiéndose 9 flujos de trabajo principales, los 6 primeros son conocidos como flujos de ingeniería y los tres últimos como flujos de apoyo.

  - Modelamiento del negocio.
  - Requerimientos.
  - Análisis y diseño.
  - Implementación.
  - Prueba (Testeo).
  - Administración del proyecto.
  - Administración de configuración y cambios.
  - Ambiente.

Flujos de trabajo de RUP (Actividades)

- **Modelamiento del negocio:** Describe los procesos de negocio, identificando quiénes participan y las actividades que requieren automatización.
- **Requerimientos:** Define qué es lo que el sistema debe hacer, para lo cual se identifican las funcionalidades requeridas y las restricciones que se imponen.
- **Análisis y diseño:** Describe cómo el sistema será realizado a partir de la funcionalidad prevista y las restricciones impuestas (requerimientos), por lo que indica con precisión lo que se debe programar.

Flujos de trabajo de RUP (Actividades)

- **Implementación:** Define cómo se organizan las clases y objetos en componentes, cuáles nodos se utilizarán y la ubicación en ellos de los componentes y la estructura de capas de la aplicación.

- **Prueba (Testeo):** Busca los defectos a lo largo del ciclo de vida.

- **Instalación:** Realiza actividades (empaque, instalación, asistencia a usuarios, etc.) para entregar el software a los usuarios finales.

Flujos de trabajo de RUP (Actividades)

- **Administración del proyecto:** Involucra actividades con las que se busca producir un producto que satisfaga las necesidades de los clientes.

- **Administración de configuración y cambios:** Describe cómo controlar los elementos producidos por todos los integrantes del equipo de proyecto en cuanto a la utilización/actualización concurrente de elementos, control de versiones, etc.

- **Ambiente:** Contiene actividades que describen los procesos y herramientas que soportarán el equipo de trabajo del proyecto, así como el procedimiento para implementar el proceso en una organización.

Mejores Practicas en RUP

Mejores Practicas en RUP

- RUP identifica 6 mejores practicas con las que define una forma efectiva de trabajar para los equipos de desarrollo de software.
  - Gestión de requisitos.
  - Desarrollo de software iterativo.
  - Desarrollo basado en componentes.
  - Modelado visual (usando UML).
  - Verificación continua de la calidad.
  - Gestión de los cambios.

Gestión de requisitos

- RUP brinda una guía para encontrar, organizar, documentar, y seguir los cambios de los requisitos funcionales y restricciones. Utiliza una notación de Caso de Uso y escenarios para representar los requisitos.

Desarrollo de software iterativo

- Desarrollo del producto mediante iteraciones con hitos bien definidos, en las cuales se repiten las actividades pero con distinto énfasis, según la fase del proyecto.

Desarrollo basado en componentes

- La creación de sistemas intensivos en software requiere dividir el sistema en componentes con interfaces bien definidas, que posteriormente serán ensamblados para generar el sistema.
- Esta característica en un proceso de desarrollo permite que el sistema se vaya creando a medida que se obtienen o se desarrollan sus componentes.

Verificación continua de la calidad

- Es importante que la calidad se evalúe en varios puntos durante el proceso de desarrollo, especialmente al final de cada iteración.
- En esta verificación las pruebas juegan un papel fundamental y se integran a lo largo de todo el proceso.

Gestión de los cambios

- El cambio es un factor de riesgo crítico en los proyectos de software.
- El software cambia no sólo debido a acciones de mantenimiento posteriores a la entrega del producto, sino que durante el proceso de desarrollo, especialmente importantes por su posible impacto, como son los cambios en los requisitos.

Gestión de los cambios

- Por otra parte, otro gran desafío que debe abordarse es la construcción de software con la participación de múltiples desarrolladores, trabajando a la vez en una release, y quizás en distintas plataformas.
- La ausencia de disciplina rápidamente conduciría al caos. La Gestión de Cambios y de Configuración es la disciplina de RUP encargada de este aspecto.

Mejores Practicas en RUP