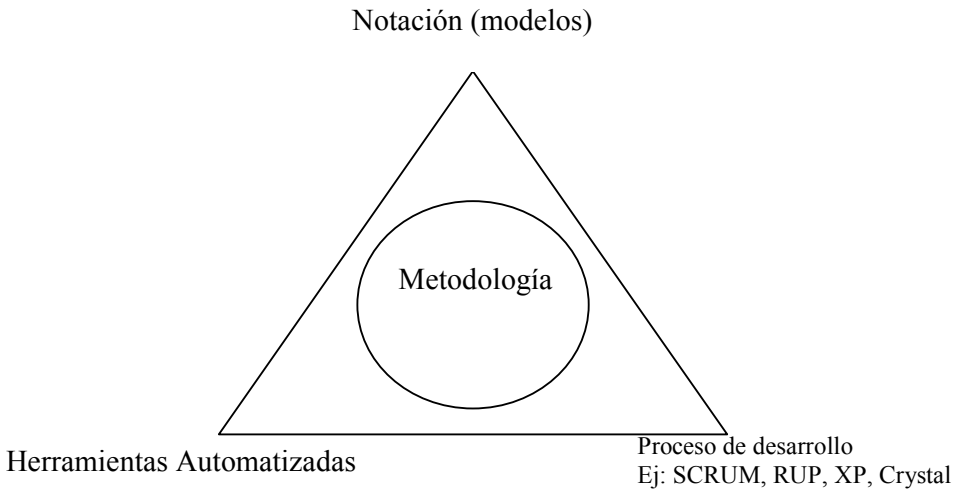
Ingeniería del software

Definición: Es un enfoque sistemático, disciplinado y cuantificable aplicado al desarrollo, la operación y el mantenimiento del software.

Para trabajar, se necesitan metodologías que requieren: notación, herramientas automatizadas y proceso de desarrollo.

Pilares de las metodologías:



La Ingeniería del software tiene 2 enfoques:
1) Estructurado
2) Orientado a Objetos

Modelos

Definición: Representación simplificada de la realidad.

Ejemplo: En ingeniería civil se necesita construir una casa.
Los pasos para construirla son los siguientes:

- Captura de requisitos
- Análisis: de acá surgen los primeros modelos.
  Ejemplo: planos esquemáticos, maquetas…
  Estos modelos sirven para poder entender el problema y comunicar a las otras personas la estructura de este problema.
  También sirven para manejar riesgos:
  - Plazo
  - Presupuesto
  - Funcionales
- Diseño: planos estructurales, hidráulicos, eléctricos
- Construcción

En los modelos de análisis tenemos que pensar en **QUE**, pero cuando pasamos a los modelos de diseño hay que pensar en el **COMO**.

Etapas de la Ingeniería del software

1) Captura de requisitos
2) Análisis
    a. Modelos de análisis (ejemplo: modelo de casos de uso orientado a análisis).
3) Diseño
    a. Modelos de clases de diseño
    b. Modelos de casos de uso de diseño
    c. Modelos de componentes.
4) Construcción
5) Pruebas
    a. Unitarias
    b. De Integración
    c. De sistema
    d. De usuarios
6) Despliegue
7) Mantenimiento
    a. Correctivo
    b. Evolutivo (ejemplo: por el dinamismo del proceso de negocio).

Proceso de Desarrollo

Ciclo de vida: El encadenamiento de las etapas se conoce como ciclo de vida.

Ciclo de vida en CASCADA

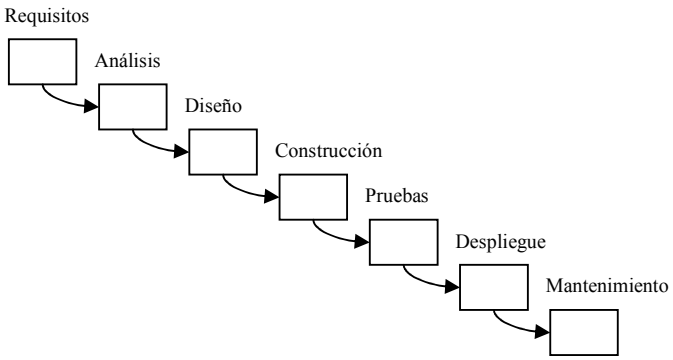
Puede servir en proyectos pequeños.

Ventajas:

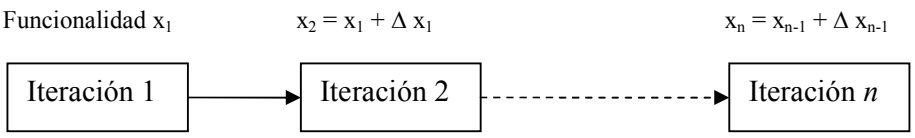
- Es simple de llevar a cabo.

Desventajas:

- No puedo paralelizar.
- No tengo entregables intermedios lo cual ayuda a detectar problemas en una etapa temprana. Si falla la captura de requisitos, hay pérdida de tiempo y plata.



Ciclo de vida ITERATIVOS e INCREMENTALES



Cada iteración está compuesta por un miniciclo en cascada.
Se debe comenzar por las funcionalidades de mayor riesgo.
Los riesgos son más manejables cuando menor sea el **ciclo de vida**.

Metodologías basadas en UML

UML: Unified Modeling Language

Definición: Es un lenguaje para visualizar, especificar, construir y documentar sistemas bajo el paradigma de orientación a objetos.

Es un lenguaje basado en modelos visuales que abarcan todo el ciclo de vida.

Estas metodologías usan distintos modelos:

	Modelo	Diagramas	
1	**Negocios**	Actividad	
2	**Casos de uso**	Casos de uso	Especificación de casos de uso
3	**Objetos**	Clases	Objetos
4	**Interacción de objetos**	Secuencia	Colaboración
5	**Estado de objetos**	Transición de estado de objetos	
6	**Componentes**	Componentes	
7	**Despliegue**	Despliegue	

Modelos

Modelo de Negocio

Sirve para entender la organización, procesos de negocios para los cuales vamos a desarrollar el sistema.

Los procesos de negocio son los que voy a querer 'automatizar' con el sistema.

Ejemplos:
1) Gestión de facturación
2) Administración de pedidos
3) Gestión de almacenes

Lo primero que hay que definir es cual es el **objetivo** de nuestro sistema.
Luego tengo que analizar cual va a ser el alcance funcional de nuestro sistema lo cual determina los límites de mi sistema.
Objetivo: Completar.

Alcance: En un principio es difuso y luego se va refinando a medida que avanza el análisis.
Actores: Todo aquello que va interactuando con nuestro sistema.
Para un diagrama de actividad hay que definir escenarios para los procesos de negocios.
Escenarios:

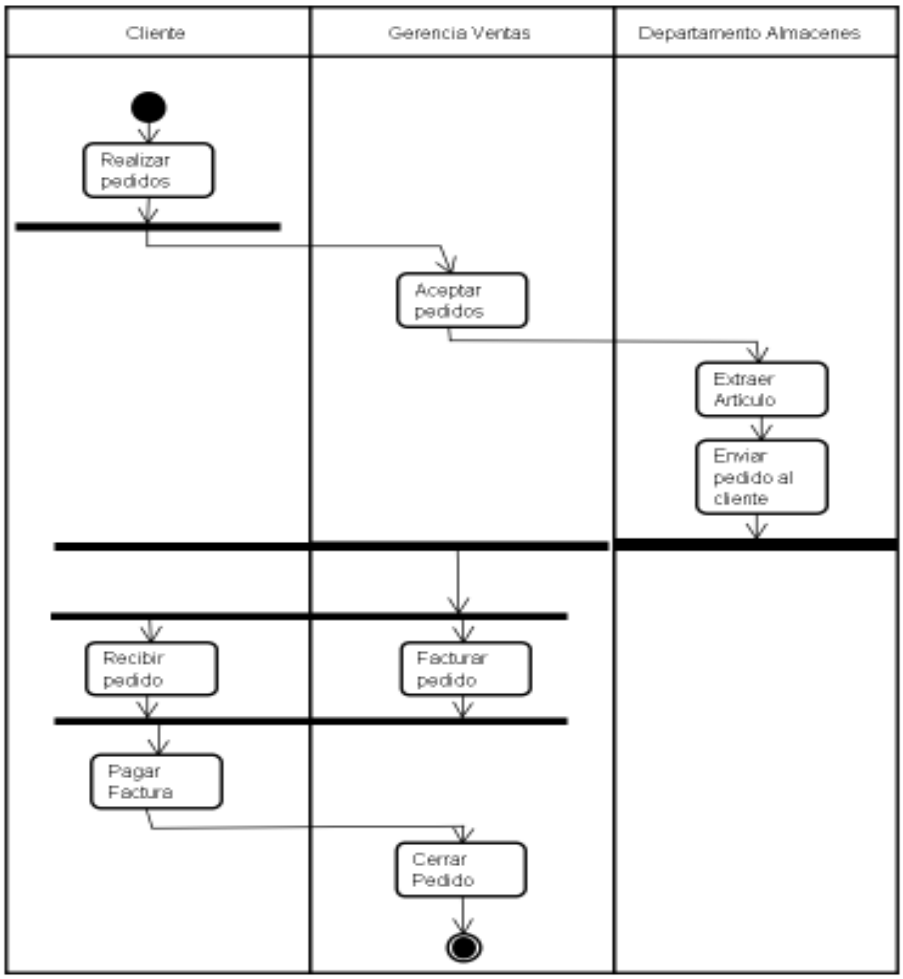
- Nivel Global: visión global de los procesos. Alto nivel de abstracción.
- Nivel Intermedio: una parte de los procesos de negocio.
- Un proceso detallado. Método.

Ejemplo: ***Nivel Global***.
Organización: Venta de productos, administrando el stock en almacenes.

- Va a tener clientes realizando pedidos.
- Gerencia de ventas
  - Administra pedidos
  - Gestiona facturación
- Departamento de almacenes.

Escenario Global:
El proceso inicia cuando un cliente realiza un pedido.
La empresa procesa dicho pedido, factura y envía el remito, finalizando cuando el cliente recibe el pedido y paga la factura correspondiente.

Pasos:
Definir escenario → Actividades → Diagrama Actividades → Modelizar secuencia actividades.

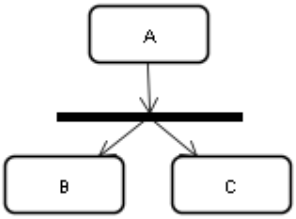


Los elementos en un diagrama de actividad son:

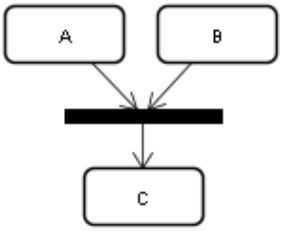
- Swimming lanes
- Actividades
- Transición de actividades
- Barras de sincronización
- Símbolo de inicio y final.

Sintaxis a utilizar: Verbo + Objeto. Describir QUE se hace (no el COMO). Las actividades tienen que hacer hincapié tal como son vistas por quienes interactúan con el sistema.

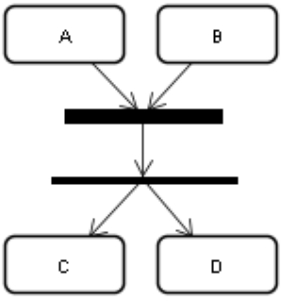
Barras de sincronización



Para que comiencen las actividades B y C, tienen que haber terminado la actividad A.



Para que comience la actividad C, tienen que haber terminado las actividades A y B.



Para que comiencen las actividades C y D, tienen que haber terminado las actividades A y B.

Modelo de Casos de Uso

Ejercicio: ***Sistema de registro de cursos.***

Al comienzo de cada semestre, los estudiantes consultan el catálogo de cursos ofrecidos en dicho semestre. Dicho catálogo contiene información de los cursos, tales como: Profesor, Departamento y Correlativas.
El sistema permitirá a los alumnos, registrarse en hasta 4 cursos en el semestre.
Cada estudiante indicará 2 alternativas de inscripción para el caso de falta de vacantes o cancelación de curso.
Ningún curso puede tener más de 20 alumnos o menos de 3 (caso en que se cancela el curso).
Finalizado el proceso de registro por un estudiante, el sistema manda información a un sistema de facturación externo para que el alumno reciba las facturas correspondientes.
Los profesores acceden para suscribirse en los cursos que quieren dictar en el semestre, también pueden consultar los alumnos anotados en su curso.
Existe un período de tiempo donde los alumnos pueden modificar su inscripción.
Durante ese tiempo el alumno puede inscribirse a nuevas materias o cancelar previas inscripciones.
El catálogo de curso de cada semestre, así como mantener actualizada la información de materia, profesores y alumnos es responsabilidad del departamento de alumnos.

Objetivo: Modelizar las principales funcionalidades provistas por el sistema.

Componentes:
- Casos de uso: Modelizar funciones requeridas por el sistema.
- Actores: Aquellos que interactúan con el sistema.
- Diagrama Casos de uso: Modeliza relaciones entre actores y casos de uso.

Actores: Algo o alguien que interactúa con nuestro sistema pero que no es parte del mismo.

Reglas prácticas para identificar actores

- ¿Quién está interesado en cierto requerimiento?
- ¿En qué lugar de la organización se usará el sistema?
- ¿Quién se beneficia usando el sistema?
- ¿Quién suministrará, usará o eliminará del sistema tal información?
- ¿Quién mantiene los datos del sistema?
- ¿Juega una persona diferentes roles?
- ¿Juegan varias personas el mismo rol? → Son el mismo actor.
- ¿Interactúa el sistema con otros sistemas?

Cuando uno define un actor debe pensar en como va a interactuar con el sistema, cual es su rol con el mismo.

Documentación de actores

Contiene una breve descripción de cada actor, la cual debe identificar el ROL que juega el actor al interactuar con el sistema.
Ejemplo:
ESTUDIANTE: *es la persona empadronada para tomar cursos en la facultad.*
PROFESOR*: persona habilitada para dictar clases en la facultad.*
DTO DE ALUMNOS*: responsable del mantenimiento del sistema de cursos.*
SIST DE FACTURACION*: sistema externo responsable de facturar los cursos a los alumnos.*

Casos de Uso

Definición: Representa o modelizar la funcionalidad provista por el sistema. El conjunto de casos de uso de un sistema define todas las formas posibles en que el sistema puede ser utilizado.

Un caso de uso especifica el comportamiento de un sistema. Es una descripción de un conjunto de secuencias de acciones, incluyendo variaciones que ejecuta un sistema para producir un resultado observable que sea de valor para un actor.

Identificación de casos de uso

- ¿Cuáles son las tareas de cada actor?
- ¿Un actor determinado, creará, modificará, eliminará o consultará información del sistema?
- ¿Qué casos de uso crearán, modificarán, eliminarán o consultarán el sistema?
- ¿Algún actor necesita informar al sistema de cambios?
- ¿Necesita algún actor ser informado de cambios que ocurran?
- ¿Qué casos de uso mantendrán actualizado el sistema?
- ¿Todos los requerimientos funcionales están cubiertos en los casos de uso?

Modelo de casos de uso

Casos de uso → Funcionalidades
Actores → Interacción con el sistema
Diagrama de casos de uso → Relaciones entre actores y casos de uso

Ejemplo:
1) Registrar curso a tomar
2) Registrar curso a dictar
3) Consultar catálogo de cursos
4) Consultar alumnos inscriptos
5) Cancelar curso
6) Mantener cursos
7) Mantener alumnos
8) Mantener profesores
9) Mantener materias

Un buen caso de uso posee completitud:
- **TEMPORAL:** completo de principio a fin.
- **FUNCIONAL**: cuando hay casos de uso que tratan con las mismas entidades.
    - Ejemplo: Crear / Modificar / Eliminar cursos → Mantener cursos.

Documentación de casos de uso

Descripción: Describe el comportamiento de casos de uso en pocas líneas con una definición de alto nivel de una funcionalidad provista por el sistema.

Flujo de eventos: Debe hacer hincapié en **QUE** debe hacer el sistema, no en el **COMO**.
Precondiciones, Flujo principal, Sub flujos, Flujos de excepción.

El flujo de eventos principal debe destacar:

1) Cuando el caso de uso comienza
2) Que interacciones se dan entre el sistema y los actores
3) La secuencia normal de eventos
4) La descripción de flujos de excepción
5) Indicar cuando el caso de uso termina

Ejemplo:

Use Case: Procesar Transacciones	
Descripción: procesa transacciones pendientes contra la cuenta bancaria del cliente	
Actores participantes: Cajero	
Pre-condiciones: existen transacciones pendientes de procesamiento	
Flujos	
Flujo Principal	
1	El actor Cajero inicia (o indica que desea iniciar) el procesamiento de transacciones pendientes
2	El sistema ordena las transacciones de modo que todas las correspondientes a una cuenta en particular se encuentren juntas y, dentro de cada grupo, se procesan primero los **depósitos** para evitar saldos negativos innecesarios.
3	Por cada cuenta: **{Determinar Cuenta de Cliente}** 3.1 El sistema determina la cuenta del cliente a la cual se aplicará la transacción. 3.2 Por cada transacción: **{Aplicar Transacción}** 3.2,1 El sistema aplica la transacción a la cuenta. Las transacciones de **depósito** incrementan el saldo de la cuenta y las de **extracción** lo disminuyen. **{Registrar Transacción}** 3.2.2 El sistema registra los datos de la transacción en una bitácora para dejar evidencia de aquella. 3.2.3 El sistema marca la transacción como completada **{Resumir Transacciones}** 3.3.3 Cuando se procesaron todas las transacciones para una cuenta determinada, el sistema genera una transacción resumen.
4	Cuando se procesaron todas las transacciones, termina el caso de uso.
Flujos Alternativos	
A1	***Cuenta Inexistente*** En **{Determinar Cuenta de Cliente}**, si la cuenta no existe:
1	El sistema marca todas las transacciones de la cuenta como **suspendidas.**
2	El procesamiento continúa en el paso siguiente a **{Determinar Cuenta de Cliente}**
A2	***Manejo de sobregiro sin autorización*** En **{Aplicar Transacción}**, si la transacción genera sobregiro (saldo negativo de la cuenta) y la cuenta no tiene autorización para ello:
1	El sistema aplica la transacción y la marca como "sobregiro"**.**
2	El sistema aplica intereses de sobregiro a la cuenta**.**
3	El procesamiento continua en **{Registrar Transacción}.**
A3	***Manejo de sobregiro con autorización*** En **{Aplicar Transacción}**, si la transacción genera sobregiro (saldo negativo de la cuenta) y la cuenta tiene autorización para ello:
1	Si la transacción no genera un sobregiro que supera el límite autorizado para el cliente, el sistema aplica la transacción**.**
2	Si la transacción genera un sobregiro que supera el límite autorizado para el cliente, el sistema aplica la transacción, la marca como "sobregiro" y aplica intereses de sobregiro a la cuenta por la cantidad que supera el límite autorizado**.**
3	El procesamiento continúa en **{Registrar Transacción}.**
Post-condiciones: todas las transacciones procesadas	

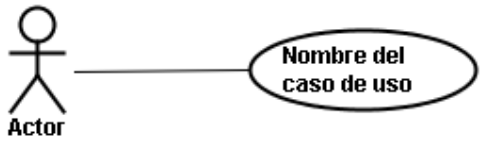
Modelo de Casos de Uso

1) Actores
2) Casos de uso
3) Relaciones
    a. Diagrama de casos de uso

Relaciones
Dos tipos:

1) **Asociación** → Aquellas que se dan entre actores y casos de uso.
2) **Dependencia** → entre casos de uso.

Asociación:



Dependencia

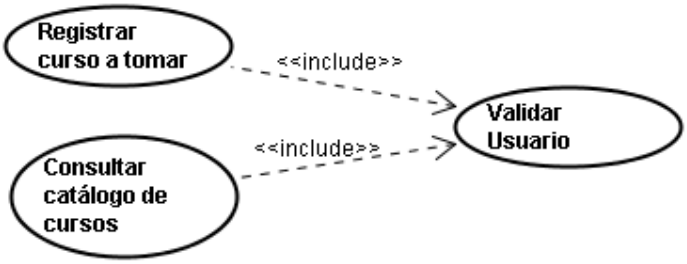
Dos tipos principales

- Inclusión: **INCLUDE**
- Extensión: **EXTEND**

Inclusión o INCLUDE

Es cuando dos o más casos de uso comparten parte de su funcionalidad, es conveniente factorizar la parte común y colocarla en un caso de uso separado a través de una relación de inclusión.

Ejemplo:



Extensión o EXTEND

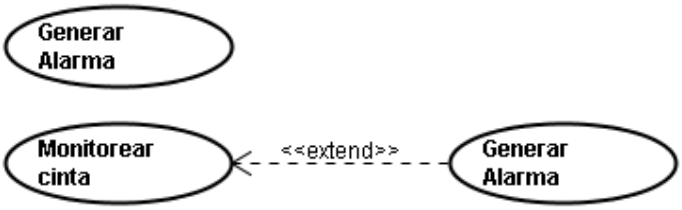
El caso de uso extendido 'extiende' las funcionalidades del caso de uso base.
En este tipo de relaciones, el caso de uso base no se entera si se ejecutan o no las acciones del caso de uso extendido.

Ejemplo:
Tenemos un sistema de monitoreo de una cinta transportadora

Caso de uso base:

Bajo ciertas condiciones vamos a tener que generar ciertas acciones que no ocurren siempre

Caso de uso extendido: este ocurre solamente bajo ciertas condiciones.

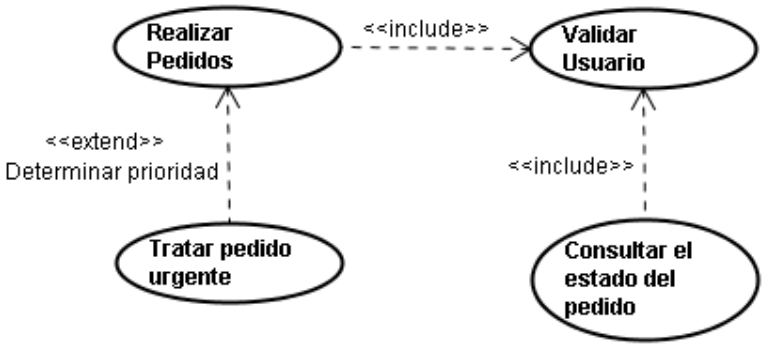


Ejemplo:

Hipótesis:

- Si la cantidad de pedidos supera los 1000 artículos y además el cliente tiene más de dos años de antigüedad con buen comportamiento de pago, entonces el pedido se realizará con prioridad urgente.
- Existirá una funcionalidad que permita consultar el estado del pedido en cualquier momento.

Caso de uso: Realizar pedido.



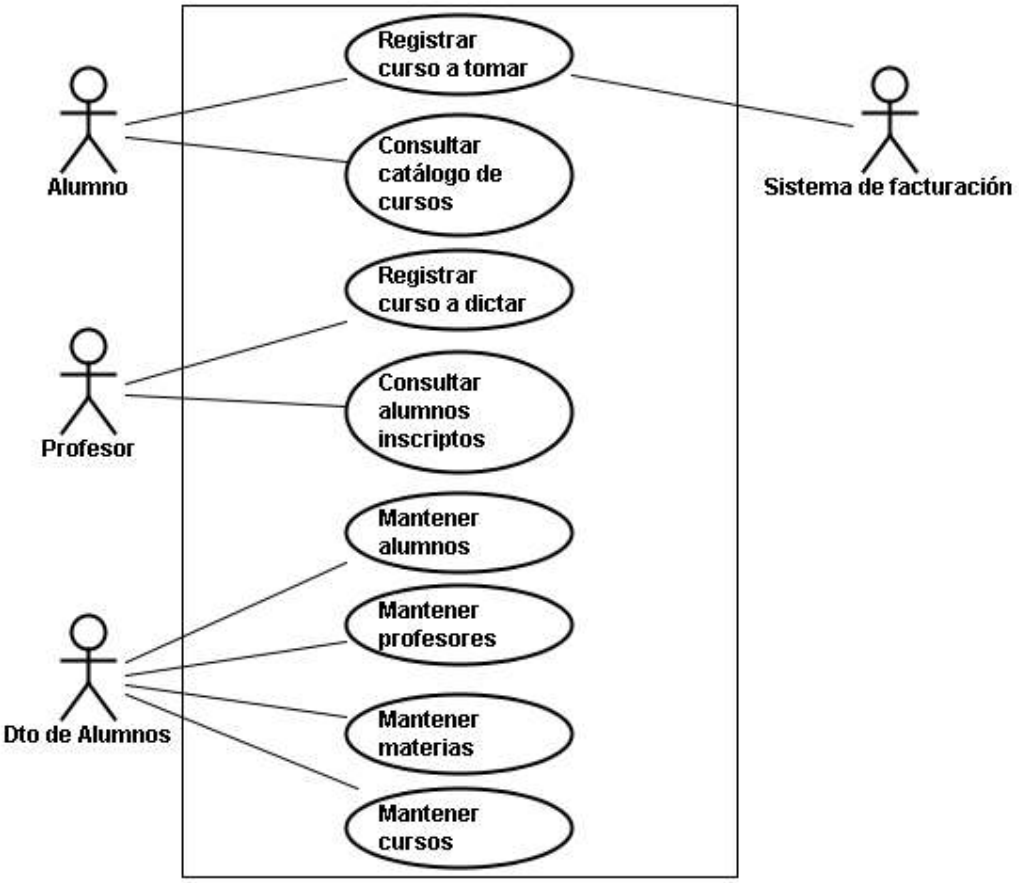
Flujo principal:

- Cliente ingresa usuario y password
- INCLUDE (validar usuario)
- Sistema solicita ingresar los ítems del pedido
- Determinar prioridad ← punto de extensión
- Sistema remite pedido y expedición

Diagrama de casos de uso para nuestro problema de registro de cursos.

Actores

1) **Primario**: el que dispara el caso de uso
2) **Secundario**: No disparan casos de uso



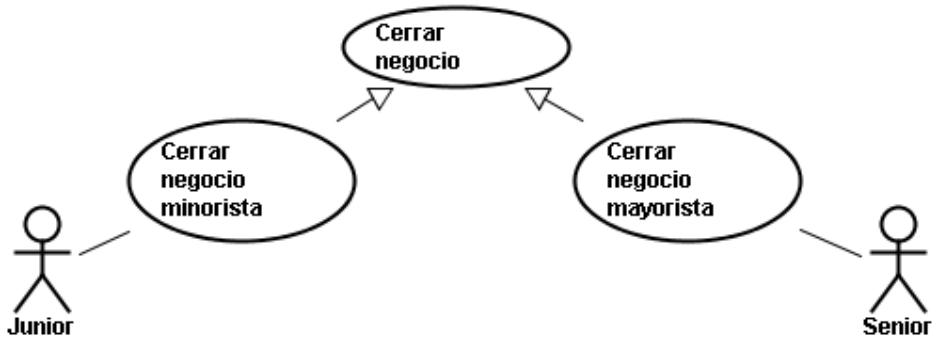
Casos especiales

Para que el departamento de alumnos observe la evolución de inscriptos, el sistema e debe emitir diariamente un informe con la cantidad de vacantes que vayan quedando para cada curso.

Jerarquías entre casos de uso

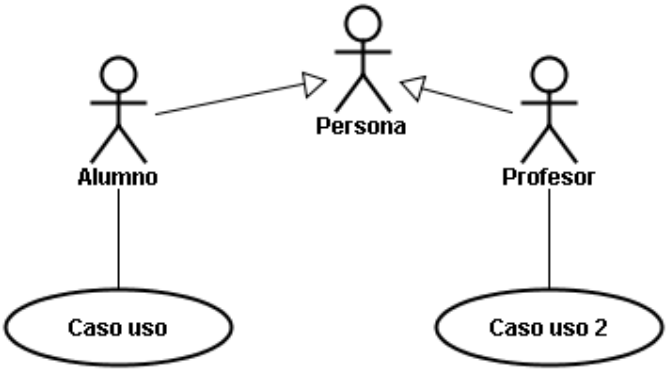
Ejemplo: Sistema para administrar ventas

Existirá una funcionalidad que le permitirá a los vendedores concretar una venta.
Existirán 2 tipos de vendedores (Junior y Senior) donde los **Junior** solamente podrán realizar ventas minoristas y los **Senior**, mayoristas.



Herencia entre actores

Relaciones jerárquicas



Ejemplo: Sistema para administrar ventas.

Cuando un cliente realiza un pedido, tendrá que llamar a un operador telefónico (call center) y la persona que lo atiende le solicitará los datos del pedido para poder ingresar los mismos en el sistema.

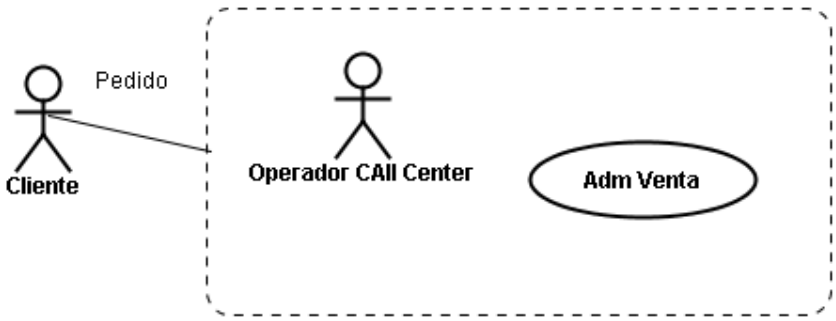
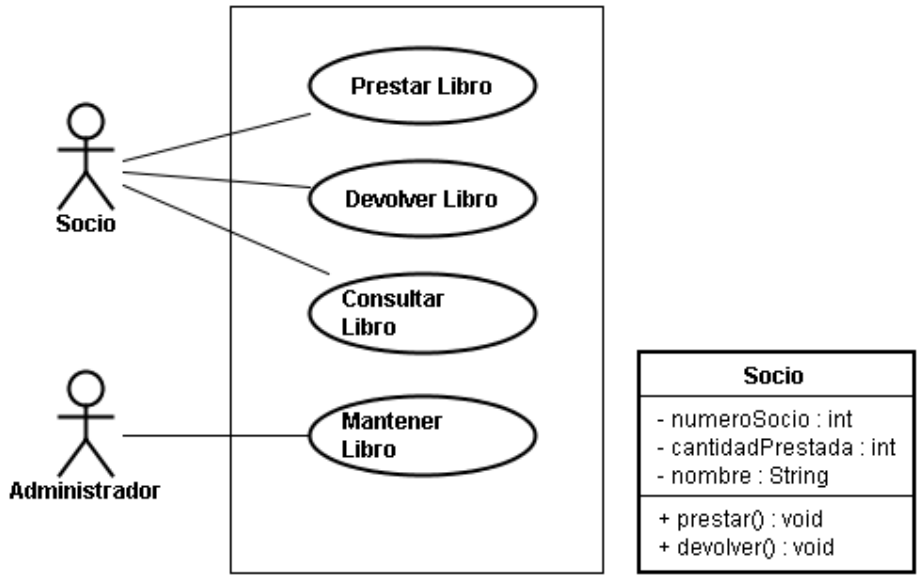


Diagrama de transición de estado de objetos

Ejemplo: Sistema de Administración de Biblioteca

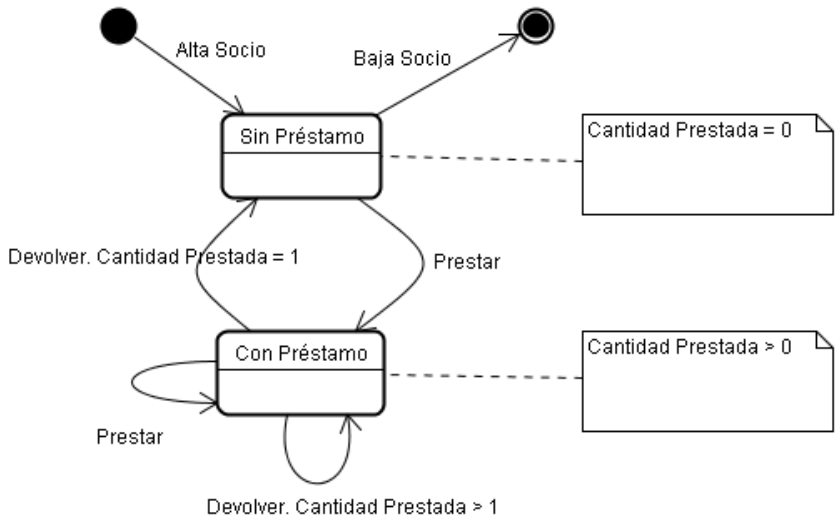


Para realizar el diagrama es necesario establecer un escenario.

El objetivo del diagrama es ver como evoluciona el estado de un determinado objeto a medida que van ocurriendo determinadas funcionalidades en el sistema.

El diagrama se basa en grafos dirigidos, los nodos indican un determinado estado del objeto y los arcos indican transición de un estado a otro.
Tiene que quedar definido tanto el estado inicial y final.
Los nodos son rectángulos con las esquinas redondeadas.

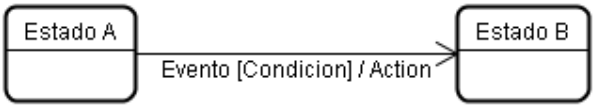
El estado inicial es sin préstamo.
El estado final con préstamo.



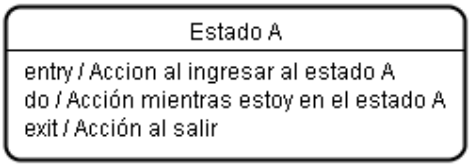
Para dar de baja un socio no puede tener libros prestados.
Un socio puede tener prestado 1 o más libros, para esto queda:

Eventos y acciones

Tenemos un objeto que pasa de un estado 'A' a un estado 'B'

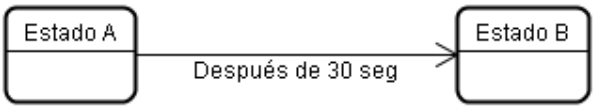


Acción solicitud de servicio a otro objeto como consecuencia de la transición.



Transacciones temporizadas

Se pasa del estado A al B, habiendo transcurrido 30 segundos se pasa al estado B.

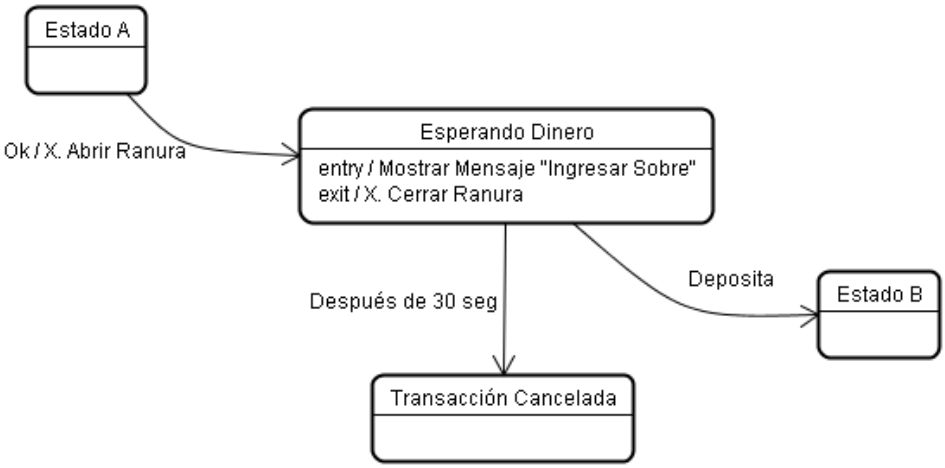


Ejemplo: Administrar las transacciones de cajero automático.

Escenario: Realizar un depósito.
Se toma una porción del escenario que comprende el ingreso del sobre con dinero.

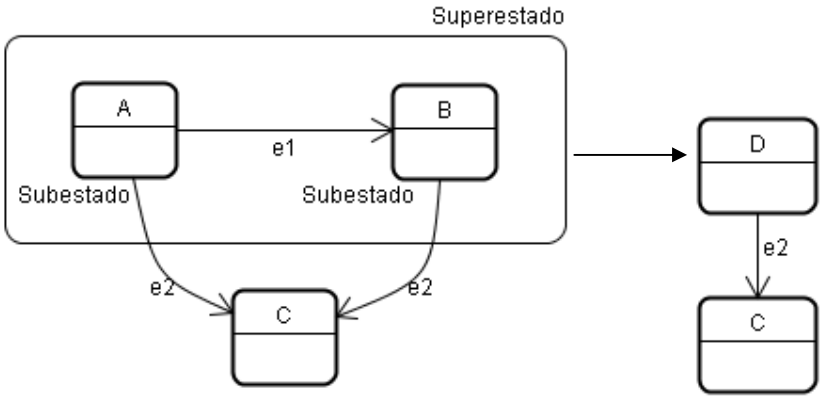
Estado A: Es el momento en el que el cliente ya puso el dinero que quiere depositar y es el momento en el cual el cajero le indica que todo está bien.

Por tiempo termino en transacción cancelada y luego de 30 segundos deposito el sobre.

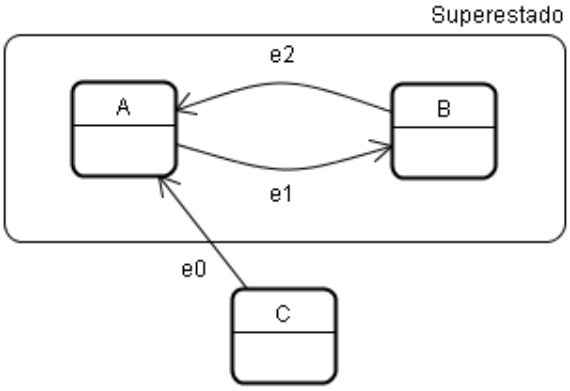


Generalización de Estados

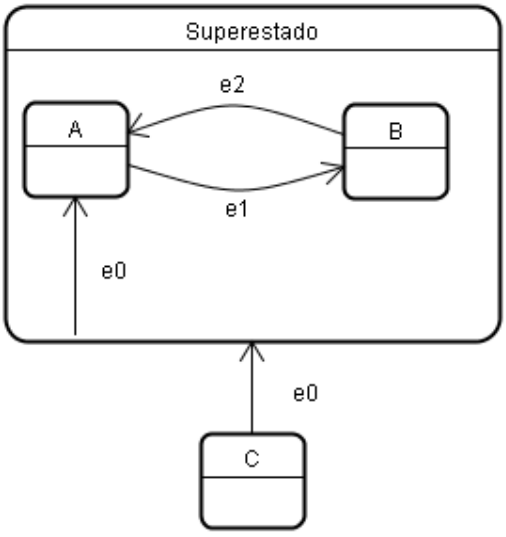
Se pasa de un superestado a un estado:



Y a la manera inversa, de:

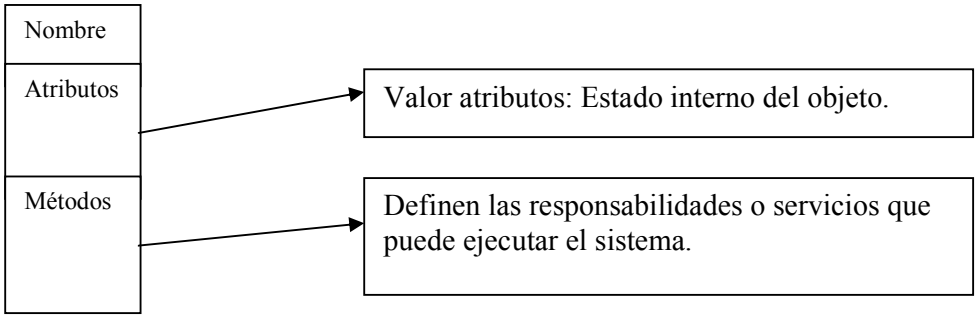


… se pasa a:



Modelo de Objetos

Diagrama de clases



Paradigma de Orientación a Objetos

1) **Objeto**: Agrupa atributos y métodos en una sola entidad.
2) **Encapsulamiento**: Valor de atributo solamente modificado a través de mensajes.
3) **Abstracción**: definición de clases.
4) **Polimorfismo**
5) **Herencia**

Clases:

Definición: Conjunto de objetos que comparten los mismos atributos, operaciones, relaciones y también la misma semántica.

La responsabilidad de una clase es un contrato u obligación que posee dicha clase.

***Modelizar clases**:*
1) Empezamos identificando clases candidatas
2) Para asegurar cuales quedan y cuales no, analizamos las responsabilidades (obligaciones que tiene que tener). Una clase debe tener por lo menos 1, pero no más de 2 o 3.

Ejemplo: Sistema de control crediticio.
Tengo la clase 'Agente Fraude', que determina el nivel de riesgo del pedido de un cliente y también maneja criterios de fraude de un cliente específico.

Identificación de clases candidatas

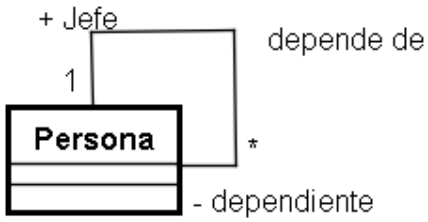
Tipos de clases:
- **Análisis**: Dominio del problema. **QUE**. Clases de entidad. Responsabilidades del sistema.
- **Diseño: COMO**.
    - **Clases de control**: Responsables de controlar secuencialidad de eventos.
    - **Clases de entorno**: modelizan interfaces de interacción entre sistema y medio.

Regla práctica:
Identificar sustantivos o clases sustantivas asociadas a las responsabilidades del sistema. (Clases de entidades candidatas).
Realizar un diagrama de clases candidatas.

Relaciones entre clases:

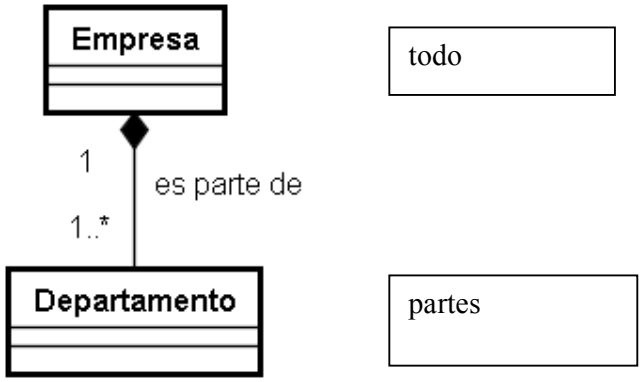
1) **Asociación**: Se usa si se necesita modelizar objetos de una clase conectándose con otras clases. Tiene que ver con la navegación.
    a. Simple
    b. Agregación
    c. Composición
2) **Generalización**: Jerarquía de clase. Herencia.
3) **Uso**: dependencia.
4) **Interfase**: relación de realización.

Clase reflexiva:



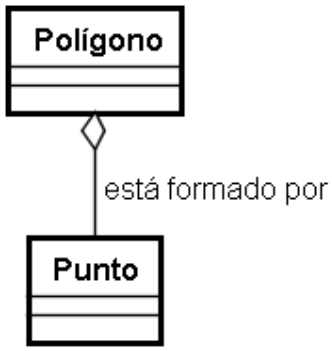
Asociaciones de composición:

La implicancia de esto es que una operación sobre este, se transmite a las clases de las cuales forman parte de. Si elimino la clase empresa, desaparece también la clase departamento.

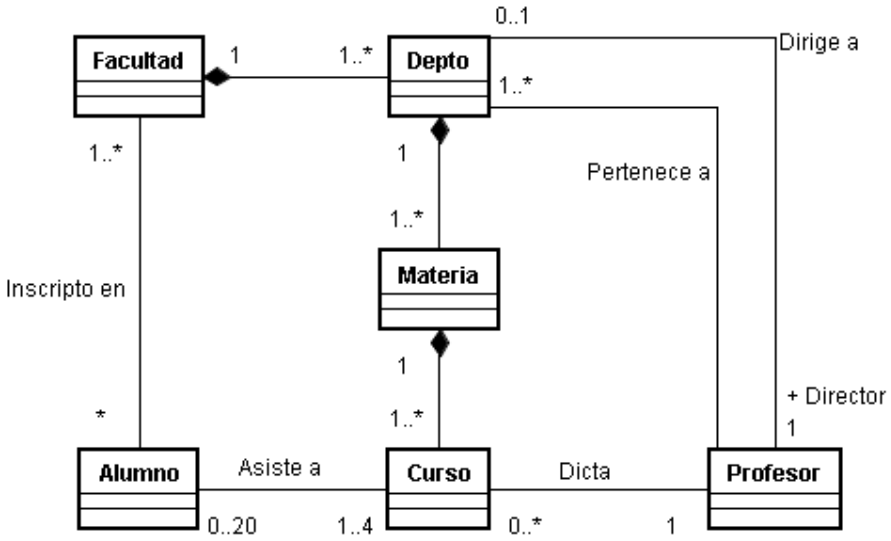


Agregación:

Los puntos tienen una jerarquía menor. Quiero expresar una condición semántica de jerarquía. 'Es parte de…', pero no es condición exclusiva.



Ejemplo Diagrama de clases: Sistema de registro de cursos.



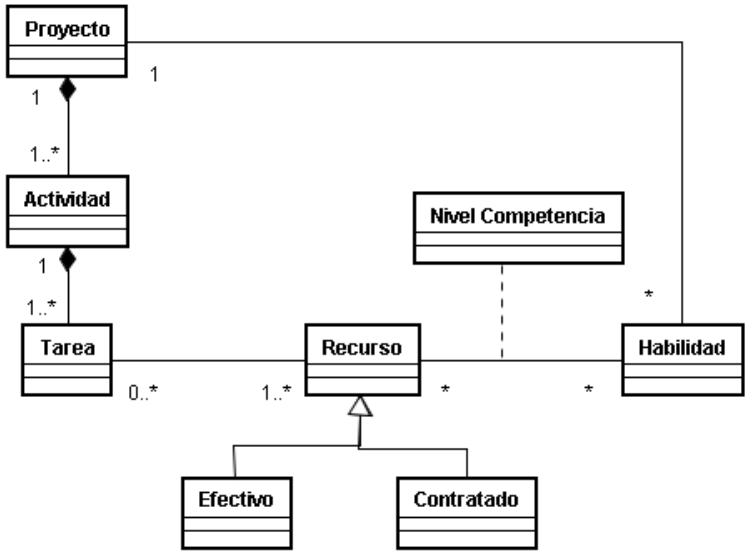
Hipótesis:

- Toda materia pertenece a un departamento
- Un profesor puede dictar materias que pertenecen a más de un departamento
- Todo departamento tiene un director el cual debe ser profesor de dicho departamento.

Ejemplo: Sistema de administración de proyectos.

Hipótesis:

- Un proyecto está compuesto de una seria de actividades (al menos 1)
- Una actividad está compuesta por una serie de tareas (al menos 1)
- A cada recurso (persona) se le asigna conjunto de tareas (puede ser 0)
- Cada recurso tiene asociado un conjunto de actividades
- Los recursos pueden ser efectivos o contratados.
- Para los proyectos se tendrán que definir habilidades que tendrán los recursos.
- Dado un recurso y una habilidad del mismo, se debe definir un nivel de competencia asociado con dicha habilidad para ese recurso.



Identificación de Clases

Se identifican clases llamadas "clases de identidad", que tienen que ver con el QUE hay que hacer y no con el COMO.

Hay 2 formas de identificarlas:

- Regla práctica
- Tarjetas CRC (Collaboration Responsability Card)

Ejemplo de tarjeta CRC

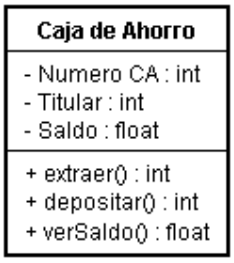
Nombre de la Clase	
Superclase	
Subclases	
Responsabilidad - -	**Colaboraciones** *(me dice que clases colaboran con esta para poder cumplir con las responsabilidades)* -

Luego se arma un diagrama de tarjetas CRC.
En una sesión de trabajo con usuarios, grupo de analistas y grupo de diseño, se refinan las clases candidatas.

Ejemplo:
Estudiar que posibilidad hay de realizar las situaciones potenciales que el cliente propone mediante las responsabilidades definidas en las tarjetas.

Sistema de Caja de Ahorro → Responsabilidades:

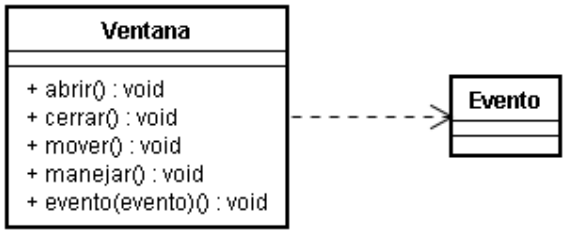
- Realizar extracciones de CA
- Realizar depósitos
- Consultar saldo



Pasos Recomendados para realizar el Diagrama de clases

1) Encontrar clases candidatas. Ver para cada clase las responsabilidades.
2) Relaciones entre clases candidatas → Diagrama de clases preliminares
3) Elaborar tarjetas CRC → Refinar clases candidatas
    a. Definir responsabilidades
    b. Definir atributos y operaciones
4) Diagrama de clases final

3. Relaciones de Uso (dependencias)

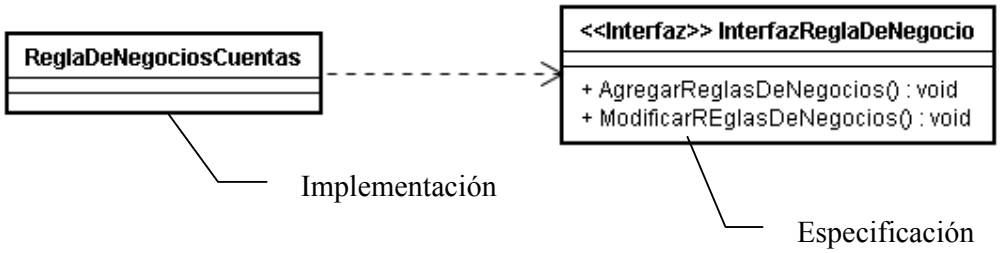


La clase ventana tiene un método que usa a la clase evento, este es un ejemplo de "Relación de uso" o "Relación de dependencia".

4. Relación de Realización (interfase)

Definición: Una interfase es una colección de operaciones que se utiliza para especificar un servicio de una clase. En definitiva declarando una interfase podemos establecer un comportamiento deseado de una abstracción independiente de la implementación de dicha abstracción.

Ejemplo: Sistema para administrar cuentas bancarias.
Reglas de negocio:



Se llama relación de realización porque en la clase realizo (implemento) lo que declaro en la interfase.
En la interfase especifico servicios y los implemento en la clase.

La forma simplificada sería:

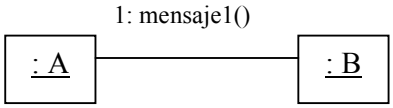
Nota: La interfaz no tiene atributos y tampoco tiene instancias.

Interacción entre objetos

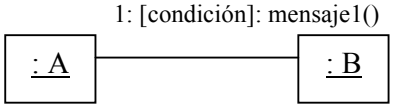
1) Diagrama de Colaboración → *Espacial*
2) Diagrama de Secuencia → *Cronológico*

Diagrama de Colaboración

Tiene objetos.



Puede tener condiciones:



Otro ejemplo:

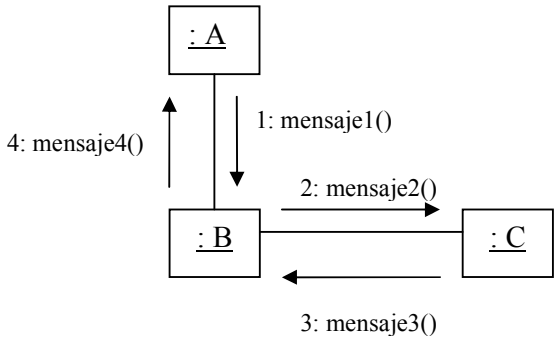
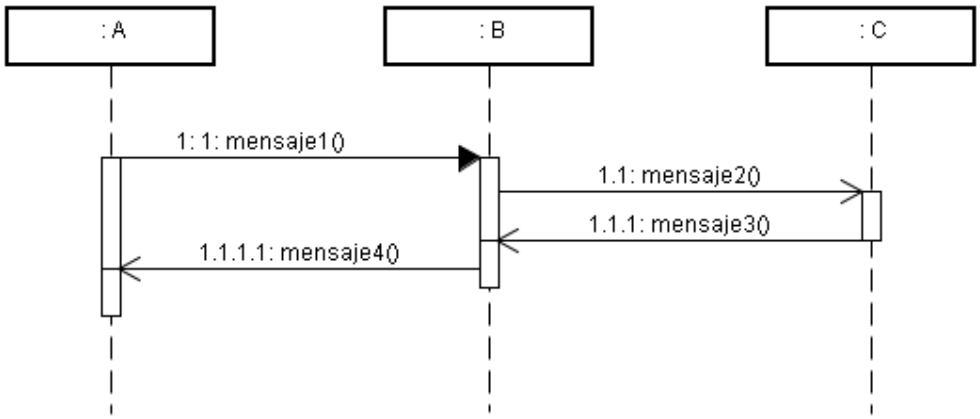


Diagrama de secuencia:

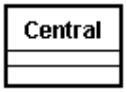


Proceso de desarrollo RUP

Diagrama de transición de estado de objetos

Este diagrama es un grafo dirigido, donde los **nodos** representan el **estado** y los **arcos**, los **eventos o transiciones**.

Ejemplo: Sistema Central Telefónico.
Escenario: Abonado 'A' llama al abonado 'B'.

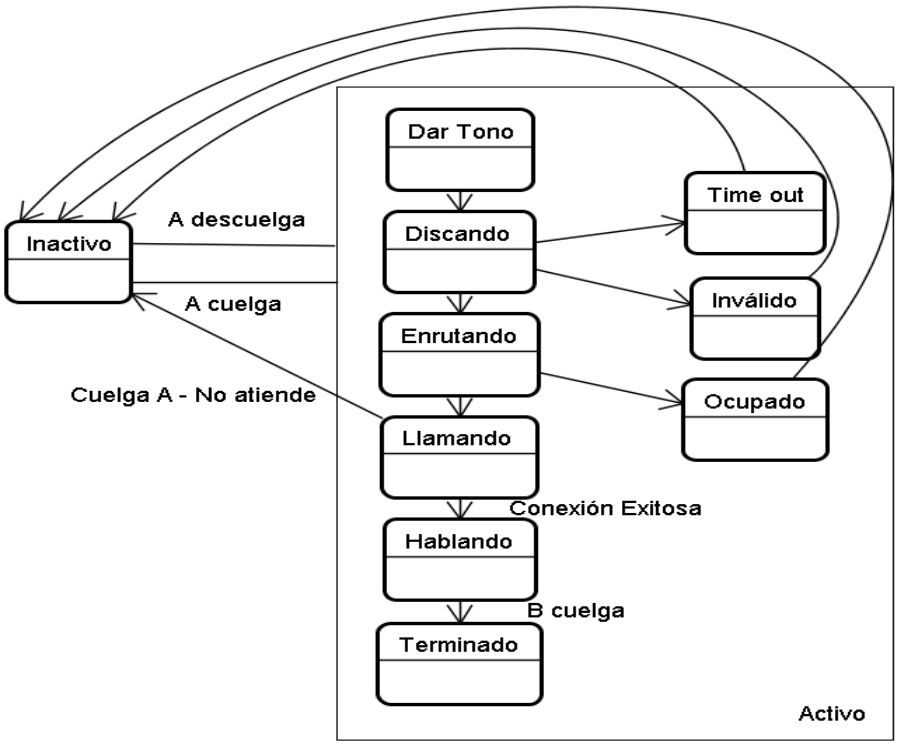


Hacer el diagrama de transición de estados para el objeto central para el escenario indicado.

Características de estado:

- Inicialmente, la central se encuentra en estado inactivo
- Cuando 'A' descuelga → Estado = **dar tono**
- Cuando 'A' discó el 1er dígito → Estado = **discando**
- Al completarse la cantidad de dígitos necesarios para llamar → Estado = **enrutando**
- Lograda la conexión → Estado = **llamando**
- Cuando 'B' descuelga → Estado = **hablando**
- Cuando 'B' cuelga → Estado = **terminado**
- En cualquier estado definido, si 'A' cuelga → Estado = **inactivo**

Hay que contemplar un estado de **time out** cuando 'A' no haya discado la cantidad de dígitos necesarios y hayan pasado más de 5 segundos antes de discar el siguiente dígito.
También contemplar estado = **inválido** si 'A' discó un dígito inválido.
Por último, considerar estado = **ocupado.**



Modelo final:

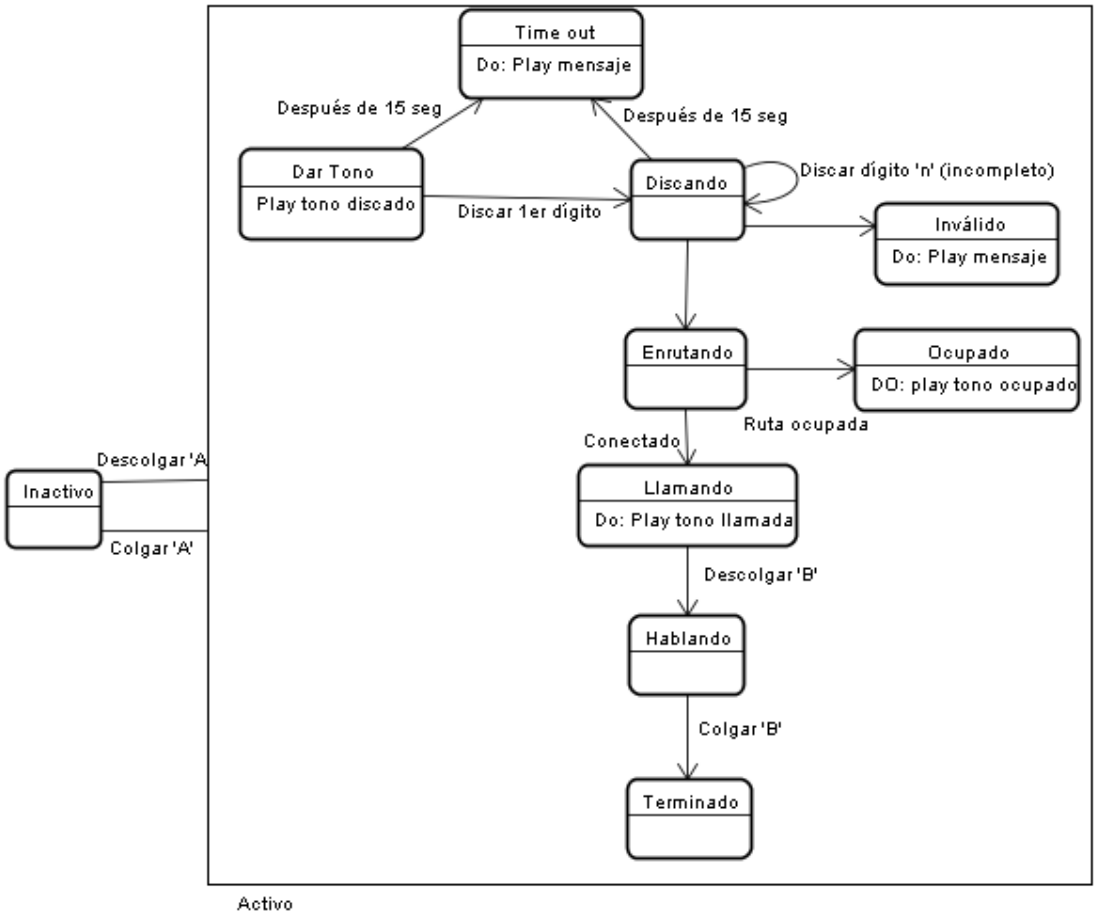
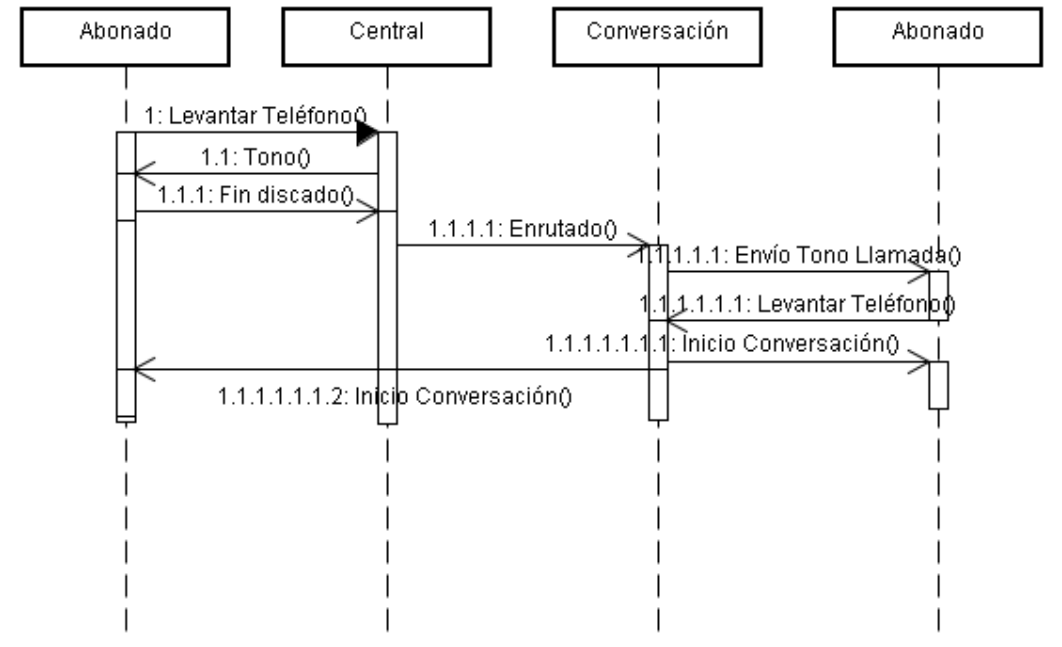


Diagrama de secuencia para el ejemplo de la Central Telefónica



Proceso de desarrollo del software

Definición: forma disciplinada de asignar tareas y responsabilidades en un proyecto de desarrollo de software.
En definitiva debe definir **QUIEN** hace que, **CUANDO** lo hace y **COMO** lo hace.

Objetivos: Asegurar la producción de software de calidad dentro de plazos y presupuestos predecibles.

Proceso de desarrollo → **RUP** ***(Rational Unified Process)***

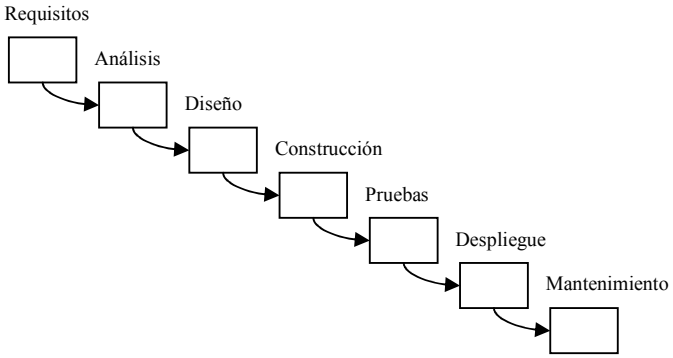
Historia:

Década 80	87/95	96/97	98
Enfoque ERICSON (Jacobson)	Objectory (Object Factory)	Agrega UML	Agrega: • Tests funcionales • Tests rendimiento • Gestión cambios • Gestión de Configuración • Modelo Negocio • Diseño de Interfases

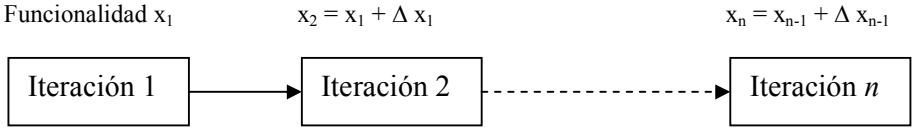
Mejores prácticas en el desarrollo de software:

1) Proceso Iterativo e Incremental
2) Proceso dirigido por casos de uso
3) Arquitecturas basadas en componentes
4) Modelización visual del software
5) Adecuada gestión de cambios
6) Verificación de la calidad del software

1)



Ciclo de vida ITERATIVOS e INCREMENTALES

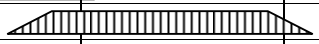
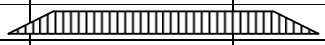


Cada iteración está compuesta por un miniciclo en cascada.
Se debe comenzar por las funcionalidades de mayor riesgo.
Los riesgos son más manejables cuando menor sea el **ciclo de vida**.

Según RUD, las fases son:

1) Inicio
2) Elaboración: Se divide en sus respectivas iteraciones, a las cuales RUD llama: **'*Componentes del proceso*'.**
   RUD define 3 componentes de soporte de proceso.
   a. Administración de Proyectos
   b. Gestión de Configuración / Cambio
   c. Gestión Ambientes / Entorno / Herramientas.
3) Construcción
4) Transición

Matriz de Esfuerzos

Componentes de Proceso	**Inicio**	**Elaboración**	**Construcción**	**Transición**
Modelo de Negocio				
Captura de Requisitos				
Análisis				
Implementación				
Pruebas				
Despliegue				
	It1 It2	**It3 It4 It5**	**It6 It7 It8**	**It9 It10**

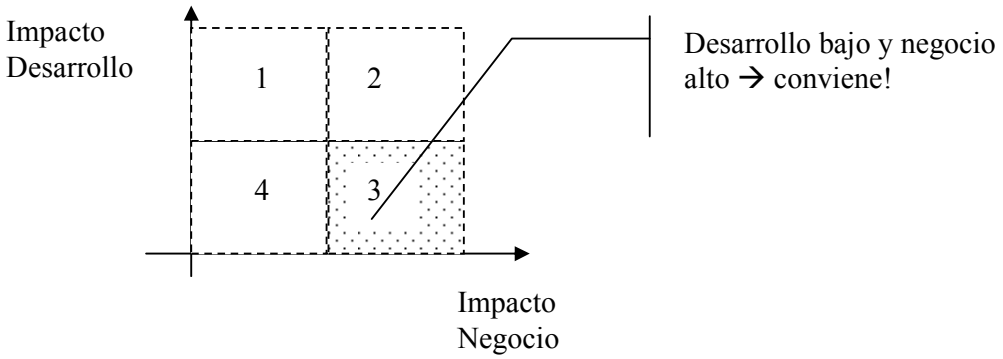
2) Los casos de uso se usan a lo largo de todo el proceso de desarrollo. Sirven para capturar y validar requisitos funcionales del sistema. Son una guía para realizarlas e implementarlas, y para validar lo construido.
Los Casos de Uso y los escenarios del proceso de desarrollo son una buena práctica para capturar requerimientos y además guiar el diseño, implementación y las pruebas.

3) Características Arquitectura
   a. Flexible
   b. Fácil de Modificar
   c. Promover Reutilización de componentes

4) Modelización Visual
   a. Estructuras
   b. Arquitectura
   c. Componentes
      Modelos – Notación Gráfica UML (base modelado visual RUP)

5) Gestión del cambio
   Los cambios son inevitables por la dinámica del negocio cambiante.
   Los procesos de negocio son cambiantes.
   Modelos de Negocio → Sistemas sufren cambios → Automatización y soporte a los procesos de negocio.

   a. Evaluación del Impacto en el desarrollo
   b. Evaluación del Impacto en el negocio.



6) Verificación Calidad del desarrollo

RUP propone que el proceso de aseguramiento de calidad tiene que ser parte del desarrollo.
No es responsabilidad de un grupo independiente en un momento determinado.
Todo grupo tiene responsabilidad a lo largo de todo el proceso.

INICIO → ELABORACIÓN → CONSTRUCCIÓN → TRANSICIÓN → FIN
Objetivos | Arquitectura estable | Producto Beta | Producto Final

FASE	HITO	CONCEPTOS INVOLUCRADOS	PRODUCTOS	COMENTARIOS
Inicio	Objetivo	1) Alcance del sistema 2) Estimación plazos y costos 3) Comprensión requerimientos 4) Oportunidad del Negocio 5) Otros productos	1 y 2: Documento, visión general. 3: Modelo Inicial casos de uso. 4: Caso de negocio 5: Plan de proyecto inicial. Identificación inicial de riesgos. Prototipos iniciales.	
Elaboración	Arquitectura del Sistema Base	1) Análisis Dominio Problema 2) Establecer Arquitectura base sólida 3) Desarrollar play proyecto 4) Mitigar factores de riesgo 5) Otros productos	1: Modelo Casos de uso final 2: Descripción arquitectura 3: Plan proyecto final 4: Lista riesgos revisada 5: Manual usuario preliminar	A partir de este hito, la arquitectura, los requisitos y los planes de desarrollo son muy estables. Hay menos riesgos, se puede planificar el resto del proyecto con menos incertidumbre Se construye arquitectura ejecutable que contemple: Casos de Uso críticos y mayores riesgos identificados.
Construcción	Capacidad Operacional Inicial	1) Se desarrollan e incorporan componentes restantes 2) Todo se prueba en profundidad 3) Enfasis en producto eficiente 4) Otros productos	1: Producto de Software con todos los componentes integrados y ejecutando en plataforma adecuada. 4: Manual usuario final. Descripción release actual	Se obtiene un producto Beta que debe decidirse si puede ponerse productivo sin mayores riesgos.
Transición	Release del producto	1) Traspasar producto desarrollado a comunidad de usuarios. 2) Pruebas beta 3) Ejecución en paralelo con sistemas antiguos 4) MIgrar datos 5) Capacitación usuarios 6) Distribución producto y despliegue		Obtener autosuficiencia de parte de los usuarios Lograr consenso para liberar productos a toda la comunidad de usuarios.

RUP

Los elementos de RUP apuntan a **quién** hace y **qué** hace.
Y define como FASES al QUIEN, QUE, COMO y CUANDO.

Elementos

1) Trabajador
2) Actividad
3) Artefacto

Trabajador 

Define comportamiento y responsabilidades de un individuo.

Comportamiento: rol que juega un individuo en un momento dado.
Ejemplo: Pepe puede jugar rol de analista, luego de diseñador.

Responsabilidad: definen que ese individuo jugando ese rol va a ser responsable de una serie de artefactos y de realizar una serie de actividades.
Ejemplo: diseñar.

Actividad 

Es una unidad de trabajo que se asigna a un trabajador.
Ejemplo: definimos una unidad de trabajo mayor a 2 hs y menor a 2 días.

1 actividad involucra a 1 trabajador.
1 actividad es un número pequeño de artefactos.

Ejemplos de actividades:

- Planificar un proyecto
- Planificar una iteración
- Encontrar casos de uso
- Revisar diseño
- Ejecutar pruebas performance

Artefactos (No tienen dibujo específico)

Definición: elementos de información que son producidos, modificados o bien utilizados a lo largo del proceso de desarrollo del software.
Los artefactos sirven como INPUT a la realización de nuevas actividades y también son el resultado de dichas actividades.

Ejemplo:

1) Un modelo es un artefacto
    a. de caso de uso
    b. de negocio
    c. de objetos
2) Un elemento del modelo
    a. 1 caso de uso
    b. 1 clase

3) Un documento
   a. de arquitectura del sistema
   b. de pronóstico financiero
4) el código fuente
5) el código ejecutable

Matriz de Asignación de Actividades

Recurso	Trabajador	Actividad
Pepe	Autor de casos de uso	Definir Casos de Uso
Juan	Diseñador de Casos de Uso	Diseñar Casos de Uso
Jose	Diseñador de Objetos	Diseñar Objetos
María	Revisar Diseño	Revisar Diseño
Richard	Arquitecto	Diseñar arquitectura

Finalmente RUP vincula todo esto en **FLUJOS DE TRABAJO**

FLUJO DE TRABAJO: es una lista de actividades, un conjunto de trabajadores y un conjunto de artefactos. En un determinado proceso, una serie de flujos de trabajo, es una secuencia de actividades que produce un determinado resultado de valor al proyecto.

Flujos de trabajo: vinculación entre Actividad, lo producido por ellas y su Input.

Metodología
1) Nada
2) Ágiles
   a. XP
   b. Scrum
   c. Crystal
3) Monumentales: exceso de documentación y falta de buen funcionamiento.

Metodologías Ágiles
1) Son **adaptables** en lugar de **predictibles:**
   La metodología tiene que ser adaptable. Quieren y necesitan del cambio.
   Predictivo → Planifico → Diseño → Construye → Planifico nuevamente.
2) Son **orientadas a las personas y NO al proceso.**
   Éxito de un proyecto, se conforma por componentes lineales de 1er orden (las personas).
   En los orientados al proceso, hay que respetar el proceso tal cual es "esto es así".
   En cambio, en los orientados a las personas, la gente tiene creatividad y prevalece por sobre el proceso.

Manifiesto de Metodologías Ágiles

4 Valores:
1) Considerar al individuo y a las interacciones del equipo de desarrollo por sobre el proceso.
2) Desarrollar software que funciona por sobre buena documentación
3) Llevar a una colaboración con el cliente más que a negociación
4) Poder responder a los cambios antes que seguir estrictamente un plan

Los 4 valores impulsan 12 (11) principios

En general:

1) Prioridad en satisfacer al cliente mediante tempranas y continuas entregas de software que aporten valor
2) Dar la bienvenida al cambio: se capturan cambios tal que nuestro cliente tenga ventajas competitivas

Proceso:

3) Entregar frecuentemente software que funcione donde la frecuencia vaya desde un par de semanas hasta un par de meses como máximo
4) La gente del negocio y los desarrolladores del producto deben trabajar juntos a lo largo del proyecto
5) Debo construir el proyecto en un entorno de individuos motivados
6) El diálogo cara a cara es el medio más eficiente y efectivo para comunicar información dentro de un equipo de desarrollo
7) Que el avance o medida principal de progreso se realiza en base al software que funciona

Equipo Desarrollo

8) La atención continua a la calidad y el buen diseño, mejoran la agilidad en un proyecto.
9) La simplicidad es esencial en el proyecto
10) Las mejores arquitecturas y diseños surgen de los equipos organizados por si mismo
    Es el primero el equipo el que decide como organizarse en base a los objetivos que se le fije a dicho equipo.
11) En intervalos regulares el propio equipo reflexiona sobre como organizarse mejor a efectos de lograr una mayor eficiencia.