# PROCESO UNIFICADO DE DESARROLLO DE SOFTWARE

*Una metodología de desarrollo de software tradicional*

Mg. Marcela Daniele

*Departamento de Computación*

*Facultad de Ciencias Exactas, Físico-Químicas y Naturales*

*Universidad Nacional de Río Cuarto*

# Proceso Unificado

¿ QUE ES EL PROCESO UNIFICADO ?

Es un proceso de desarrollo de software

define ***quién*** está haciendo

***qué***, ***cuándo***, y ***cómo***

para construir o mejorar un producto de software

# Proceso Unificado

- Es una guía para todos los participantes del proyecto: clientes, usuarios, desarrolladores, directivos.
- Ordena las actividades del equipo.
- Dirige tareas de cada desarrollador como un todo.
- Especifica los artefactos a desarrollar en cada etapa.
- Ofrece criterios para el control y la medición de cada actividad y del producto resultante.
- Reduce riesgos y hace el proyecto más predecible.
- Es capaz de evolucionar junto a las tecnologías, herramientas, personas y patrones de organización.

# Proceso Unificado
## Principales Características

**Dirigido por Casos de Uso**

Centrado en la Arquitectura

Iterativo e Incremental

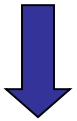
### Dirigido por Casos de Uso

Los casos de uso dirigen todo el proceso de desarrollo debido a que las actividades o flujos de trabajo: análisis, diseño, implementación y prueba, se desarrollan a partir de ellos.

**"Un Caso de Uso es una funcionalidad de un sistema que brinda un resultado de valor y observable para un actor"**

# Proceso Unificado
## Principales Características

Dirigido por Casos de Uso



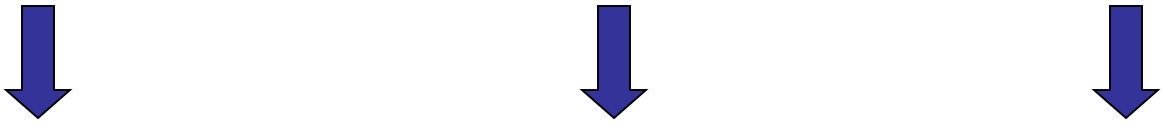
**Centrado en la Arquitectura**

Iterativo e Incremental

**Centrado en la Arquitectura** (diferentes vistas del sistema). Cada producto tiene una función (CU) y una forma (A).

- Involucra definir la plataforma: sistema operativo, sistema de gestión de base de datos, los protocolos de red.
- La arquitectura se va formando con la especificación en detalle de cada CU, en término de clases, subsistemas, componentes, hasta que es considerada suficiente.
- Requisitos no funcionales: rendimiento, fiabilidad.

# Proceso Unificado
## Principales Características



Dirigido por Casos de Uso

Centrado en la Arquitectura

**Iterativo e Incremental**

### Iterativo e Incremental

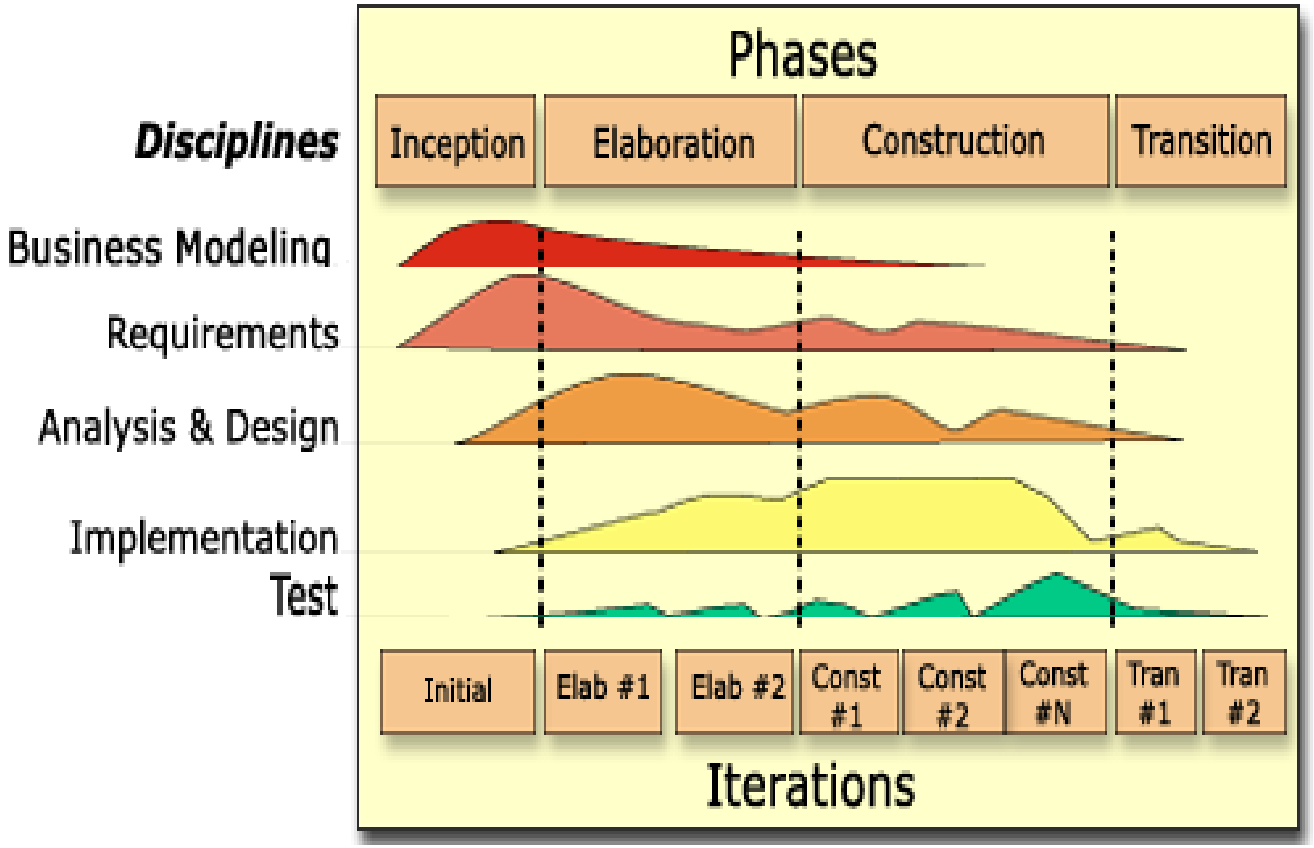
Se divide el trabajo en **miniproyectos**.

Cada uno es una ***Iteración*** que resulta en un ***incremento***.

*Iteración*: pasos en el flujo de trabajo.

*Incremento*: crecimiento del producto.

# Proceso Unificado
## Fases y Flujos de Trabajo – Ciclo de Vida




Tiempo: representa los aspectos dinámicos del proceso. Cómo se organiza y lleva adelante el proceso de desarrollo.

Componentes del Proceso: representa los aspectos estáticos del proceso: actividades, workflows, artifacts, workers, herramientas.

# Proceso Unificado
## FASES

### Inicio

- Establecer las reglas de negocio.
- Delimitar el proyecto.
- Identificar actores y casos de uso.
- Decidir si se llevará adelante o no el proyecto.

### Elaboración

- Establecer la arquitectura.
- Evalúar riesgos (requerimientos, tecnológicos, políticos).
- Especificar en detalle los CU.
- Al final de esta etapa, planificar actividades y estimar los recursos.

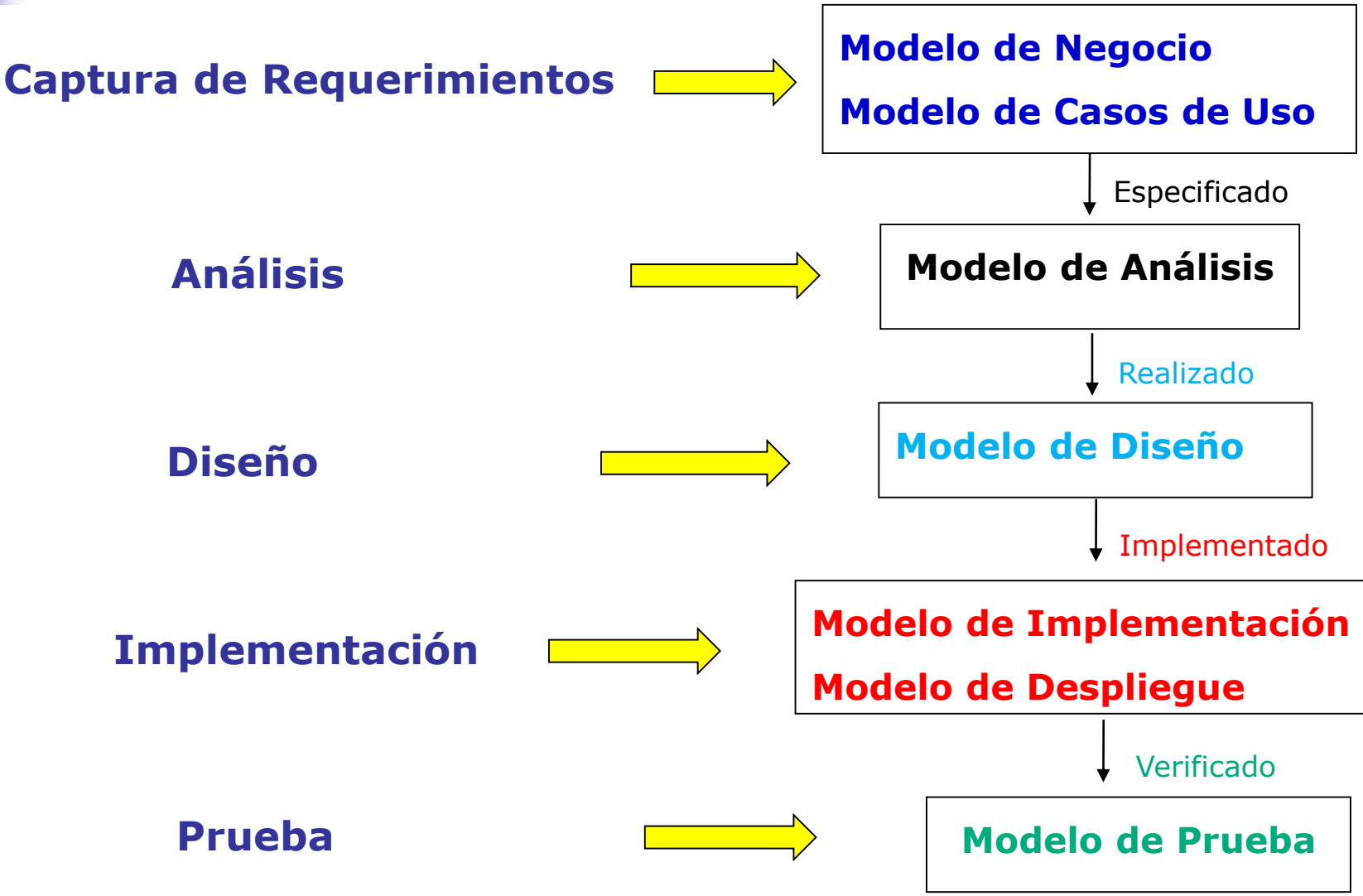
### Construcción

- Iterativa e incrementalmente se desarrolla el producto completo.
- El producto contiene los casos de uso que los desarrolladores y el cliente asintieron al desarrollo para esta versión.

### Transición

- Versión Beta del producto.
- Prueba del producto y reporte de defectos y deficiencias.

# Proceso Unificado
## FLUJOS DE TRABAJO



# Proceso Unificado
## FLUJOS DE TRABAJO

**Modelo de Negocio**: establece una abstracción de la organización. Permite comprender los procesos de negocio de la organización.

**Modelo de Casos de Uso**: incluye CU del sistema y relaciones con actores.

**Modelo de Análisis**: especificación detallada de cada CU.

**Modelo de Diseño**: define los CU como colaboraciones entre subsistemas, clases e interfaces.

**Modelo de Implementación**: realiza los CU del diseño (vista lógica) a componentes del sistema (vista física).

**Modelo de Despliegue**: modela nodos físicos y componentes del sistema.

**Modelo de Pruebas:** se especifican los casos de prueba para verificar la interacción entre componentes del sistema, el comportamiento observable externamente.

# Proceso Unificado

**Workflow (Flujo de Trabajo o Etapa)**: Captura de requerimientos, análisis, diseño, implementación y Prueba.

**Workers (Trabajadores o roles)**: roles específicos que cumplen cada equipo o cada uno de sus integrantes. (Ej: especificador de casos de uso, arquitecto, programadores, ingeniero de pruebas)

**Stakeholders**: incluye cualquier otro participante del proyecto: autoridades, usuarios, vendedores, directores de proyecto, agencias reguladoras.

# Proceso Unificado

**Artefactos**: información creada, producida o cambiada por los workers en el desarrollo del sistema.

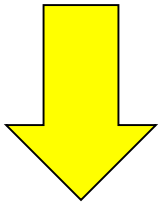
- Ingenieriles: diagramas UML, esquemas usuario-interface, prototipos, planos de prueba, procedimientos de prueba.

- Administración: generalmente tienen poca duración. Ej: Casos de negocio, plan para ubicar personas como workers.

**Actividad**: es una pieza de trabajo que un worker ejecuta en un workflow.

# Proceso Unificado

## Ingeniería de Requerimientos en PU



## Etapa: CAPTURA DE REQUERIMIENTOS

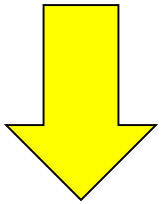
- Modelo de Negocio
- Modelo de Casos de Uso

# Proceso Unificado
## Etapa: **Captura de Requerimientos**

- Objetivo principal: Describir **QUE** hará el sistema.
- Permitir a desarrolladores y clientes un entendimiento común y acordar con la descripción de las funcionalidades del sistema.
- En la Captura de Requerimientos se construyen dos artefactos:
  - **MODELO DE NEGOCIO:** estudia el negocio sobre el que se estudiará el problema y se diseñará el sistema.
  - **MODELO DE CASOS DE USO:** especifica y describe las funcionalidades del sistema.
- Se definen requerimientos suplementarios, no funcionales.

# Proceso Unificado

## Ingeniería de Requerimientos en PU



Etapa: CAPTURA DE REQUERIMIENTOS

- **Modelo de Negocio**

# Captura de Requerimientos → Modelo de Negocio

- Entender la estructura y la dinámica de la organización.
- Entender los problemas corrientes e identificar potenciales o posibles mejoras.
- Asegurar a los clientes, usuarios, y desarrolladores que tienen una comprensión común de la organización.
- Derivar los requerimientos del sistema, necesarios para soportar la estructura y dinámica de la organización.

## ARTEFACTOS DEL MODELO DE NEGOCIO

- GLOSARIO DE TERMINOS DEL NEGOCIO
- REGLAS DEL NEGOCIO
- MODELO DE CASOS DE USO DEL NEGOCIO
- MODELO DEL DOMINIO

# PU: MODELO DE NEGOCIO

## ARTEFACTO → Glosario de Términos del Negocio

Define los términos más importantes usados por el negocio. La definición dada a cada término debe ser única y consistente durante todo el proyecto.

Nombre_Término: descripción textual

Ejemplo: **Negocio → Video Club**

**Cliente**: persona que alquiló por lo menos una copia.

**Vip**: cliente que alquila más de 30 copias de películas por mes.

**Película**: identifica las características de cada película que esta en el stock del video club, del cual puede haber más de una copia.

# PU: MODELO DE NEGOCIO

## ARTEFACTO → Reglas del Negocio

Especifican políticas o condiciones que restringen el comportamiento y la estructura del negocio.

- Definen una restricción o invariante que el negocio debe satisfacer.
- Se describen en lenguaje natural o en lenguaje formal. Generalmente, UML utiliza un lenguaje de restricción de objetos denominado OCL (Object Constraint Language).

Ejemplo: **Negocio → Video Club**

Regla 1: un cliente no puede alquilar más de 3 copias de películas por vez.

Regla 2: el video no puede tener más de 1000 clientes.

Regla 3: la devolución debe realizarse dentro de las 24 hs. a partir de la fecha del alquiler, excepto los clientes Vip que tienen 48 hs.

# PU: MODELO DE NEGOCIO

## ARTEFACTO → Modelo de Casos de Uso de Negocio

- Modela las principales funciones o procesos del Negocio.
- Su principal objetivo es ayudar al logro de un mejor y claro entendimiento del contexto del negocio, al que posiblemente se le desarrollará e implementará un sistema de software.
- Facilita la identificación de roles y responsabilidades de los miembros del negocio.

**ARTEFACTOS DEL MODELO DE CASOS DE USO DE NEGOCIO**

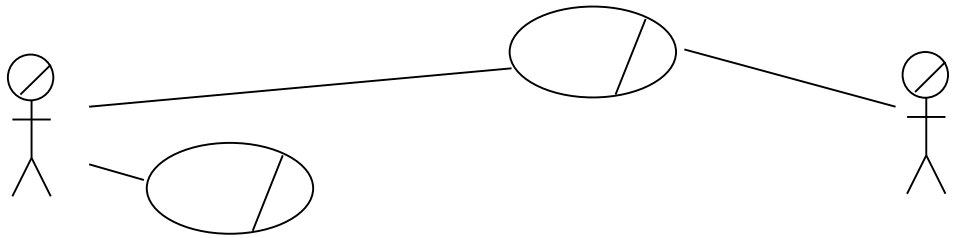
- **CASOS DE USO O PROCESOS DEL NEGOCIO**: representa una secuencia de acciones que el negocio desarrolla y por el cual obtiene un resultado de valor para uno o más actores del negocio. Permite conocer el comportamiento del negocio, el valor que provee y cómo interactúa con su ambiente.
- **ACTORES DEL NEGOCIO**: rol que algo o alguien cumple cuando interactúa con el negocio.

# PU: MODELO DE NEGOCIO

## ARTEFACTO → Modelo de Casos de Uso de Negocio

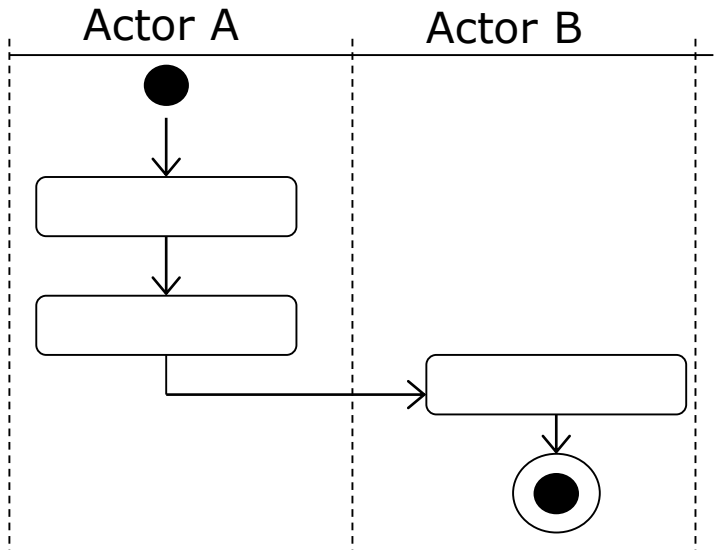
### MODELADO de CUN CON UML

- El **Diagrama de Casos de Uso de UML**, permite modelar los Casos de Uso de Negocio y sus actores.



Se agrega una línea cruzada sobre el caso de uso y el actor para indicar que se están modelando las funcionalidades del negocio y NO las del sistema.

- El **Diagrama de Actividades**, permite modelar la secuencia de acciones de cada Caso de Uso de Negocio y sus actores.



# PU: MODELO DE NEGOCIO - Ejemplo

## Modelo de Casos de Uso de Negocio del Video Club

Casos de Uso del Negocio:

- Alquilar Película
- Designar Cliente Vip

Actores del Negocio:

- Encargado
- Secretario
- Cliente

Diagrama de Casos de Uso del Negocio

Descripción del CUN: Alquilar Película

El cliente solicita alquilar una o más películas, el vendedor busca su ficha y agrega los datos de cada película. El cliente pasa por la caja y abona al encargado del video el importe correspondiente, el cual le entrega las copias alquiladas.

(Debería tener en cuenta que no puede llevar más de tres películas por vez, según regla 1)

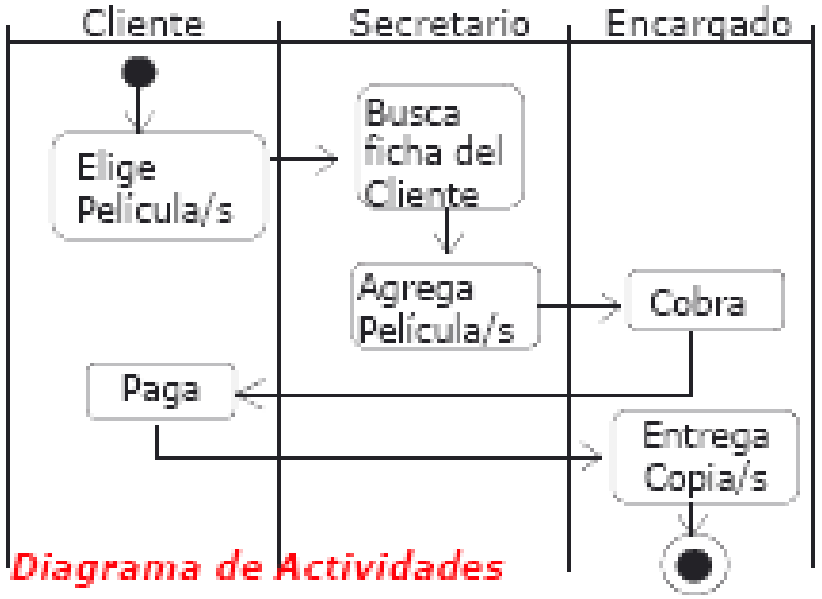


Diagrama de Actividades

# PU: MODELO DE NEGOCIO

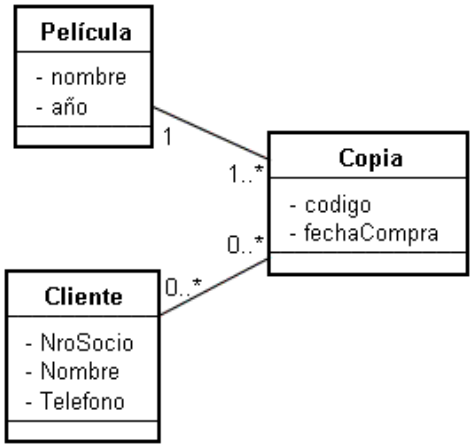
## ARTEFACTO → Modelo de Dominio

El objetivo del modelo de dominio es comprender y describir las entidades más importantes del sistema.

- Permite establecer el contexto del sistema.
- Entidad del Negocio: representa algo que los workers inspeccionan, manipulan, producen o utilizan en un caso de uso de negocio.
- Modelo de Dominio + Glosario de Términos: ayudan a lograr un vocabulario y entendimiento común del problema a resolver.

En UML, el modelo de dominio se grafica con un Diagrama de Clases

Ejemplo: Modelo de dominio del Video Club



# Proceso Unificado
## Captura de Requerimientos

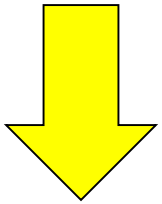
Hasta aquí se estudió y modeló el NEGOCIO, para proponer el diseño y desarrollo de un SISTEMA INFORMATICO basado en CU

**MODELO DE NEGOCIO**

**MODELO DE CASOS DE USO DEL SISTEMA**

- De Actores del Negocio a **Encontrar los Actores del Sistema**.
- Del modelo de dominio, las reglas del negocio, los procesos o casos de uso del negocio y del glosario de términos, a **Encontrar los Casos de Uso del Sistema**

## Ingeniería de Requerimientos en PU



Etapa: CAPTURA DE REQUERIMIENTOS

- **Modelo de Casos de Uso del Sistema**

# Proceso Unificado

## Captura de Requerimientos → Modelo de Casos de Uso

- MODELO DE CASOS DE USO = CASOS DE USO + ACTORES
- El Modelo de Casos de Uso, permite a clientes y desarrolladores acordar en los requerimientos del sistema.
- Es usado y refinado en las sucesivas etapas del desarrollo del sistema. (CR, Análisis, Diseño, Implementación, Prueba).
- Los casos de uso representan las funcionalidades o comportamiento del sistema, los actores representan los usuarios del mismo.

**ARTEFACTOS DEL MODELO DE CASOS DE USO**

- CASO DE USO: funcionalidad que brinda algún resultado de valor para uno o más actores.
- ACTOR: usuario que requiere del uso de las funciones del sistema.

# Proceso Unificado

## WORKERS → Modelo de Casos de Uso

- ANALISTA DE SISTEMAS

  Responsable del conjunto de requisitos (funcionales y no funcionales) modelados en los casos de uso.

  Responsable de encontrar los casos de uso y los actores, y asegurar de que el modelo es completo y consistente.

- ESPECIFICADOR DE CASOS DE USO

  Responsable de la descripción detallada de cada caso de uso.

- ARQUITECTO

  Responsable de obtener y mantener una vista de la arquitectura del sistema.

# Proceso Unificado

## ACTIVIDADES → Modelo de Casos de Uso

### ➢ ENCONTRAR ACTORES

- ❖ Determinar todos los participantes que interactuarán con el sistema.
- ❖ A partir del Modelo de Casos de Uso del Negocio, identificar un actor del sistema por cada actor del negocio que usará el sistema, luego refinar.

### ➢ ENCONTRAR CASOS DE USO

- ❖ Identificar las funcionalidades del sistema, a partir de: **modelo de dominio**, **casos de uso del negocio**, **reglas de negocio**, **glosario de términos**, y otros requisitos solicitados. Determinar los Casos de Uso.
- ❖ Factorizar los CU. Definir relaciones de include, extend y generalización.
- ❖ Modelar con un Diagrama de Casos de Uso de UML.

### ➢ PRIORIZAR CASOS DE USO

- ❖ Definir qué CU se desarrollarán en la primera iteración, y cuales se dejarán para las siguientes iteraciones.

### ➢ DETALLAR CASOS DE USO

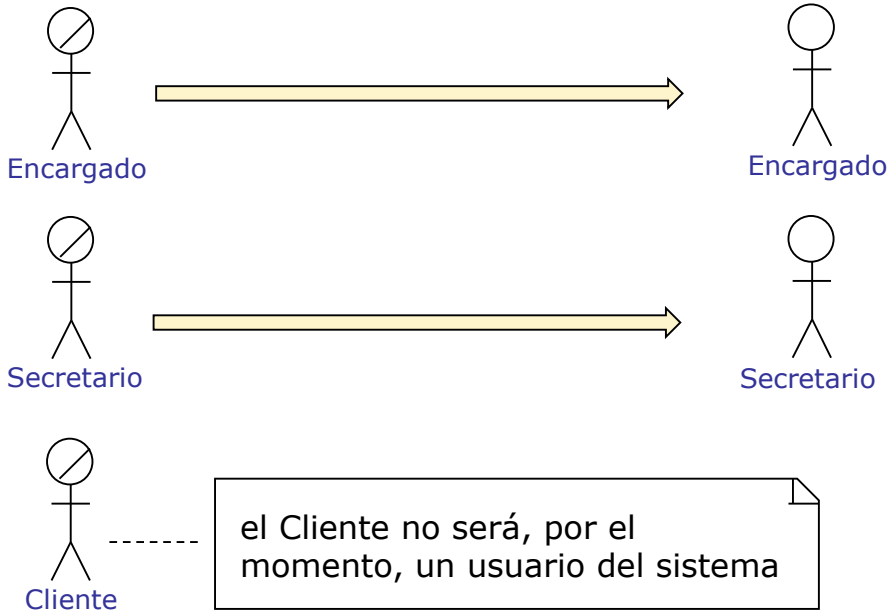
- ❖ Describir el flujo de sucesos o eventos principal y alternativo de cada caso de uso.

# Proceso Unificado

## Ejemplo → Modelo de Casos de Uso del Video Club

- Encontrar Actores

| **Actores del Negocio** | → | **Actores del Sistema** |
|---|---|---|
| Encargado | | Encargado |
| Secretario | | Secretario |
| Cliente | | |

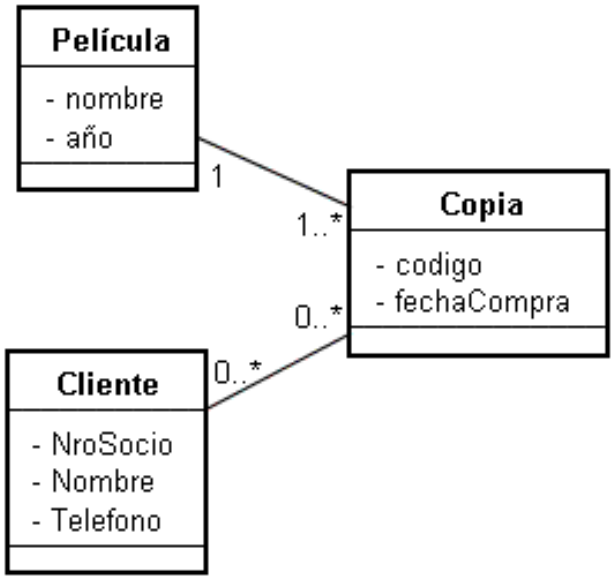


# Proceso Unificado

## Ejemplo → Modelo de Casos de Uso del Video Club

- Encontrar Casos de Uso

**Modelo de Dominio**

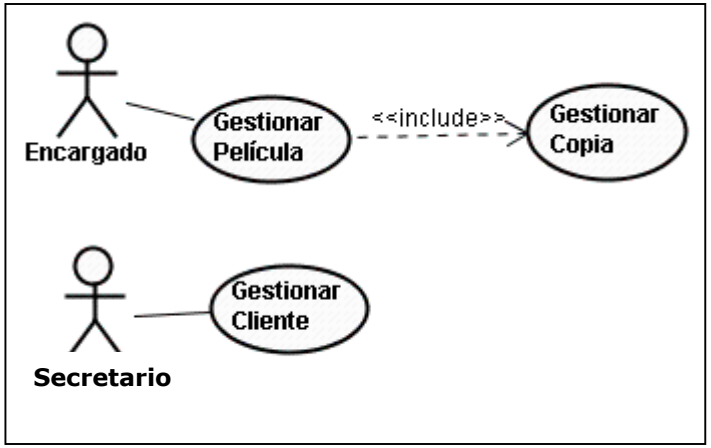



Generalmente, se obtienen los casos de uso de inserción, eliminación, modificación y búsqueda de cada entidad, modelada con una clase.

(se conocen como Gestionar o ABM)

Los Casos de Uso identificados a partir del Modelo del Dominio son:

- **Gestionar Película**
- **Gestionar Copia de Película**
- **Gestionar Cliente**



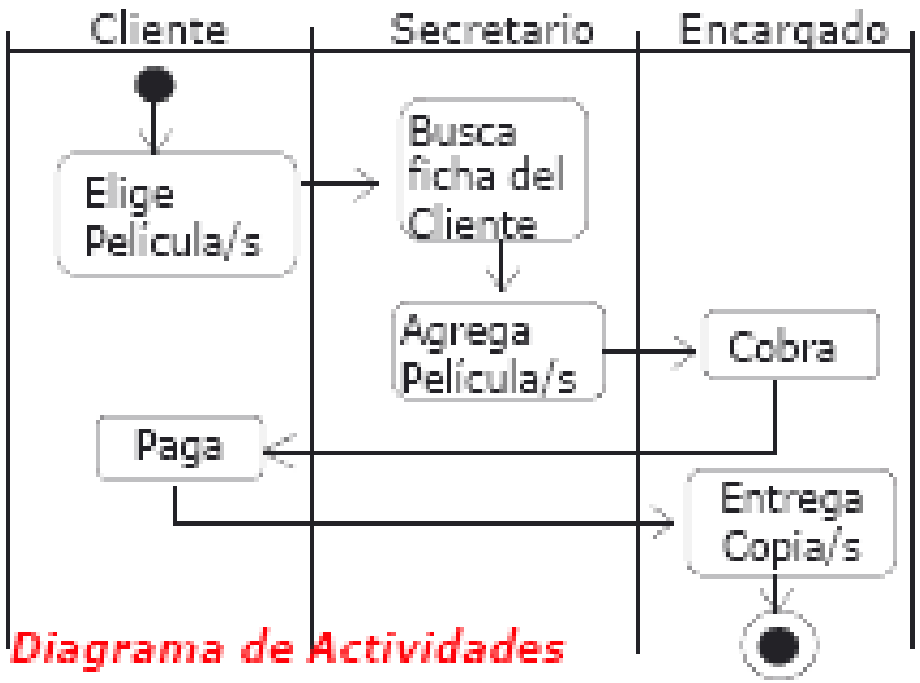
# Proceso Unificado

## Ejemplo → Modelo de Casos de Uso del Video Club

### ❖ Encontrar Casos de Uso

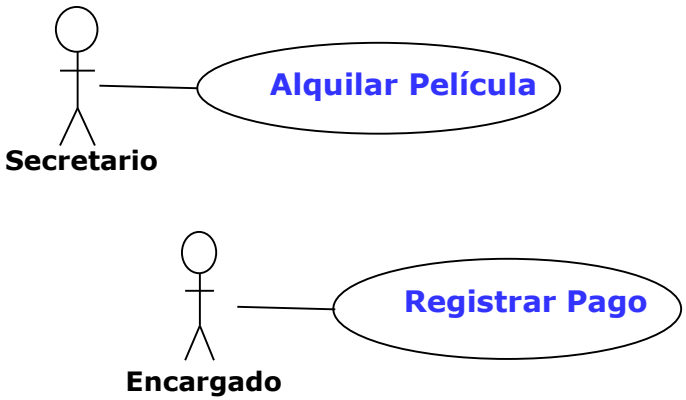
Se obtienen casos de uso analizando cada una de las actividades del **Caso de Uso de Negocio**, teniendo en cuenta el **Actor** responsable.

**CUN: Alquilar Película**




Los Casos de Uso del sistema identificados a partir del **CUN Alquilar Película** son:

- **Alquilar Película**
- **Registrar Pago**



# Proceso Unificado

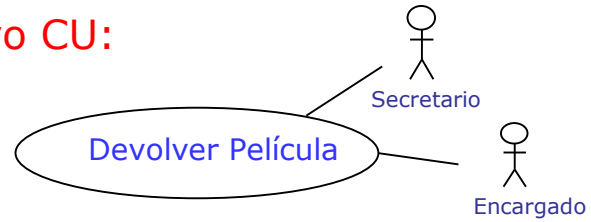
## Ejemplo → Modelo de Casos de Uso del Video Club

### ❖ Encontrar Casos de Uso

**Reglas del Negocio** ➡ Las reglas pueden imponer condiciones en los CU ya identificados o generar nuevos CU.

La **regla 1** (un cliente no puede alquilar más de 3 copias de películas por vez) impone una condición en el **CU: Alquilar Película**.

- La regla 3 permite encontrar un nuevo CU:
  - **Devolver Película**




**Glosario de Términos** ➡

Los definición de los términos debe ser consistente durante todo el desarrollo. Por ej.:

- La definición dada a "cliente" debe ser tenida en cuenta en el **CU Gestionar Clientes**.
- La definición dada a "Vip" puede generar un cambio de estado en los objetos clientes cuando cumplan dicho requisito.

# Proceso Unificado
## ACTIVIDAD → Detallar Casos de Uso

Se propone usar esta plantilla para la descripción de cada caso de uso.

| Nombre del Caso de Uso: | |
|---|---|
| PreCondición: | |
| PostCondición: | |
| Descripción: | |
| FLUJO DE EVENTOS PRINCIPAL | |
| **Actor** | **Sistema** |
| 1. .....<br>4. ..... | 2. .....<br>3. .....<br>5. ..... |
| FLUJO DE EVENTOS ALTERNATIVO | |
| | 2.1. .....<br>2.2. .....<br>5.1. ..... |

Otros Casos de Uso del sistema pueden ser definidos a partir de requerimientos del cliente, de los objetivos del sistema, del analista que identifica CU con valor agregado al negocio, etc.

# Proceso Unificado

## Etapa: ANALISIS

## MODELO DE ANALISIS

- Mapear descripción de CUS a clases de análisis
- Relacionar clases de Análisis → Diag. de Clases
- Definir escenarios →Diagrama de Colaboración

# PU – ANALISIS

| MODELO DE CASOS DE USO | MODELO DE ANALISIS |
|---|---|
| Se describe utilizando el LENGUAJE DEL CLIENTE | Se describe utilizando el LENGUAJE DEL DESARROLLADOR |
| Vista EXTERNA del Sistema | Vista INTERNA del Sistema |
| Estructurado por Casos de Uso | Estructurado por Clases y Paquetes |
| Se usa primariamente como un contrato entre el cliente y desarrolladores, determina QUE hará el sistema y se decide si el sistema se llevará a cabo o no | Se utiliza principalmente por los desarrolladores para entender QUE hará el sistema y plantear interacciones que comiencen a definir el COMO |
| Puede contener redundancias e inconsistencias | Se reduden considerablemente las redundancias e inconsistencias |
| Captura las funcionalidades del sistema | Refina y describe la forma en que se realizarán las funcionalidades del sistema |
| Define los CASOS DE USO y a futuro serán analizados en el Modelo de Análisis | Define REALIZACIONES DE CASOS DE USO, donde cada uno representa el análisis de un caso de uso particular |

# El Análisis en el Ciclo de Vida del Software

- Los requerimientos son refinados con el fin de estructurar todo el sistema.
- El propósito es lograr un entendimiento más preciso de los requerimientos a través de una descripción detallada y utilizando el lenguaje del desarrollador.
- Se definen clases y sus responsabilidades para facilitar la comprensión y estructuración del sistema.
- El foco principal del Análisis está en las primeras iteraciones de la fase de elaboración.
- Contribuye a logro de una arquitectura estable en la siguiente etapa.
- Los requerimientos son entendidos, y el análisis sirve como base para el diseño y la implementación.

# ARTEFACTOS - ANALISIS

- MODELO DE ANALISIS: jerarquía de PAQUETES DE ANÁLISIS conteniendo CLASES DE ANÁLISIS y REALIZACIONES DE CU.
  - CLASES DE ANALISIS: representan una abstracción de una o varias clases y/o subsistemas en el diseño del sistema.
  - REALIZACIONES DE CASOS DE USO: descripción de la realización de cada caso de uso en término de clases de análisis y la interacción entre objetos de análisis.
  - PAQUETES DE ANALISIS: permiten organizar el modelo de análisis en piezas más manejables. Pueden contener realizaciones de casos de uso, clases de análisis y otros paquetes de análisis.
- DESCRIPCIÓN DE LA ARQUITECTURA: vista arquitectónica del Modelo de Análisis mostrando sus artefactos más significativos.

# WORKERS - ANALISIS

- ARQUITECTO: responsable de la integridad del modelo de Análisis y de crear una vista arquitectónica que permita avanzar en el desarrollo.

- INGENIERO DE CASOS DE USO: responsable de la integridad de las realizaciones de casos de uso.

- INGENIERO DE COMPONENTES: responsable de la integridad de clases de análisis y sus relaciones, y la integridad de paquetes de análisis.

# ACTIVIDADES - ANALISIS

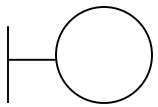
- ANALIZAR CADA CASO DE USO
  - Identificar Clases de Análisis → UML- Diagrama de Clases
  - Describir la interacción entre objetos de análisis → UML -Diagrama de Comunicación, descripción Textual
  - Capturar requerimientos especiales
- ANALISIS ARQUITECTONICO
  - Realizar un bosquejo del modelo de análisis identificando los paquetes de análisis, clases de análisis y requerimientos especiales más importantes.
- ANALIZAR PAQUETE
  - Agrupar clases en paquetes cuando es necesario.
  - Asegurar que cada paquete de análisis es lo más independiente posible de otros.
  - Asegurar que cada paquete de análisis satisface su propósito de realización de ciertas clases y casos de uso.

# ACTIVIDADES - ANALISIS

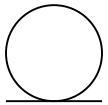
- ANALIZAR CADA CLASE DE ANALISIS
  - Asignar un nombre, Identificar Atributos, Identificar relaciones.
  - Identificar Responsabilidades: compilación de todos los roles que la clase juega en todas las realizaciones de casos de uso.
  - Capturar requerimientos especiales.

## TIPOS DE CLASES DE ANALISIS

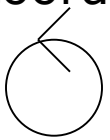
- CLASES LIMITE (Boundary class): modelan interacción entre el sistema y sus actores.



- CLASES ENTIDAD (Entity class): modelan información persistente.



- CLASES CONTROL (Control class): representan coordinación, secuenciamiento, transacciones y control de objetos.



# Ejemplo

## Obtener Clases de Análisis a partir de la descripción de un CU

Caso de Uso: **Insertar Cliente**
Precondición: existe un cliente para ser ingresado al sistema.
Poscondición: se registra el nuevo cliente en el sistema o el cliente ya estaba cargado.

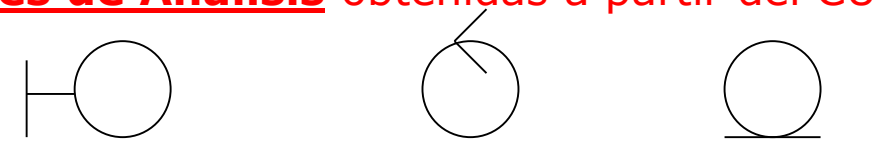
**Flujo de Eventos Principal**

| Secretaria | Sistema |
|---|---|
| 1. Ingresa el DNI del cliente. | |
| | 2. Chequea existencia. |
| 3. Ingresa nro socio, nombre, teléfono y localidad | |
| | 4. Include (buscar localidad) |
| | 5. Carga al nuevo cliente. |

**Flujo de Eventos Alternativo**

2.1. El cliente ya existe, el sistema informa tal situación y termina el caso de uso.
4.1. La localidad ingresada no existe, debe ingresar una localidad existente.

Las **Clases de Análisis** obtenidas a partir del CU Insertar Cliente pueden ser:



**IU_Cliente** **Ctrl_Cliente** **Cliente**

# Ejemplo

## Diagrama de Clases de Análisis y Diagrama de Colaboración

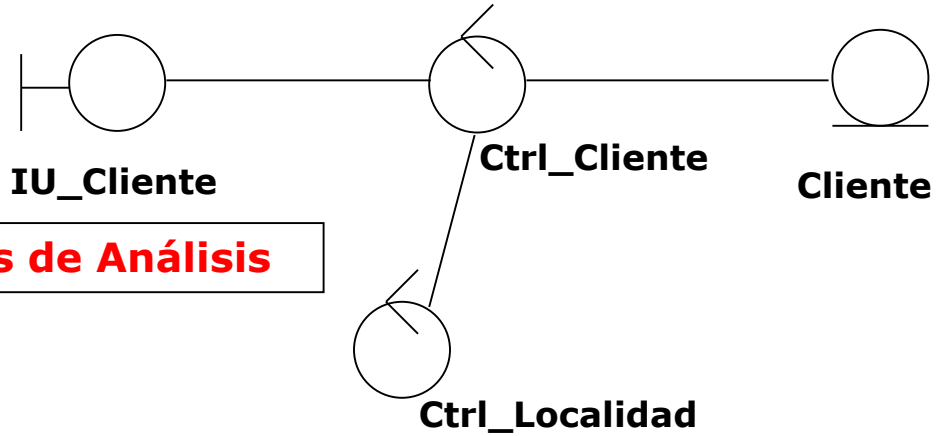
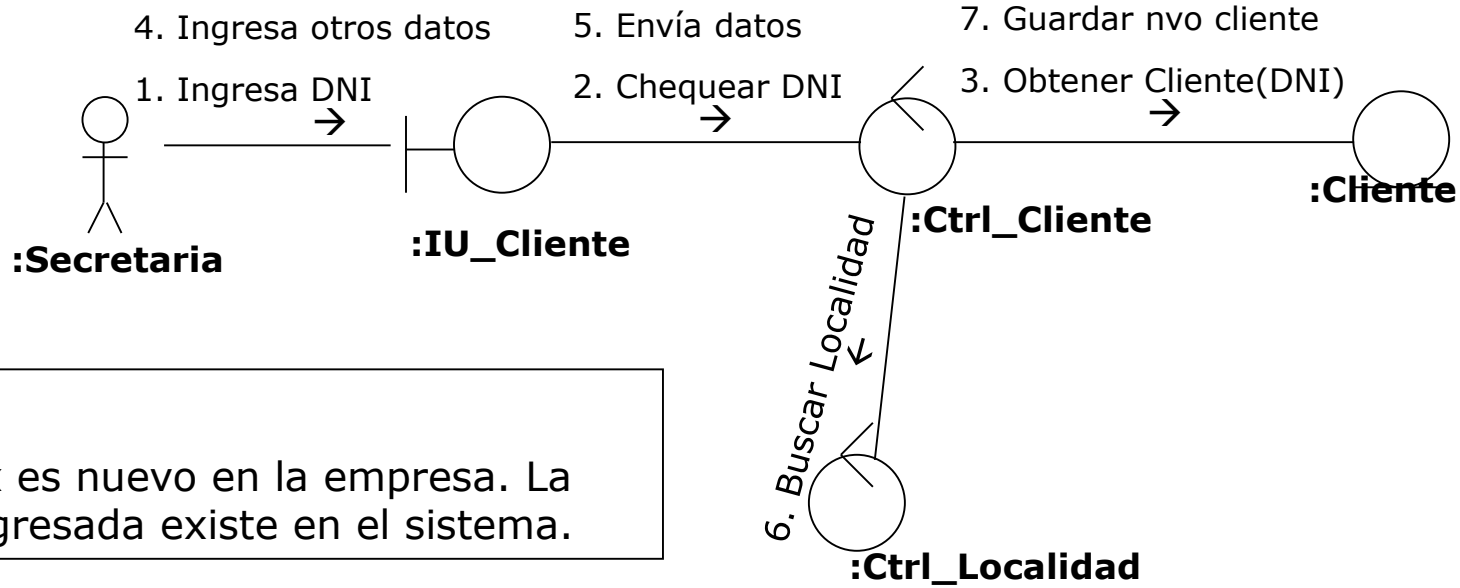



**Diagrama de Clases de Análisis**

**Diagrama de Colaboración**

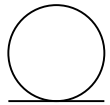



**Escenario:**

El cliente xx es nuevo en la empresa. La localidad ingresada existe en el sistema.

# Ejemplo

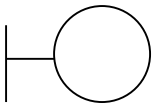
## Atributos y Responsabilidades de las Clases de Análisis



**Cliente**

Atributos: DNI, nombre, teléfono, nrosocio. Localidad??

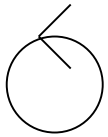
Responsabilidades:brindar y mantener guardada los datos del cliente. (set y get).



**IU_Cliente**

Atributos: ??.

Responsabilidades: recibir los datos ingresados por el usuario. Chequear formato de los datos ingresados. Mostrar resultados al usuario.



**Ctrl_Cliente**

Atributos: ??.

Responsabilidades: chequear la existencia del cliente, enviar datos para ser guardados, buscar la localidad ingresada.

# Proceso Unificado

## Etapa: DISEÑO
## MODELO DE DISEÑO

- Mapear clases de análisis a clases de diseño
- Relacionar clases de Diseño → Diag. de Clases
- Modelar escenarios → Diagrama de Secuencia

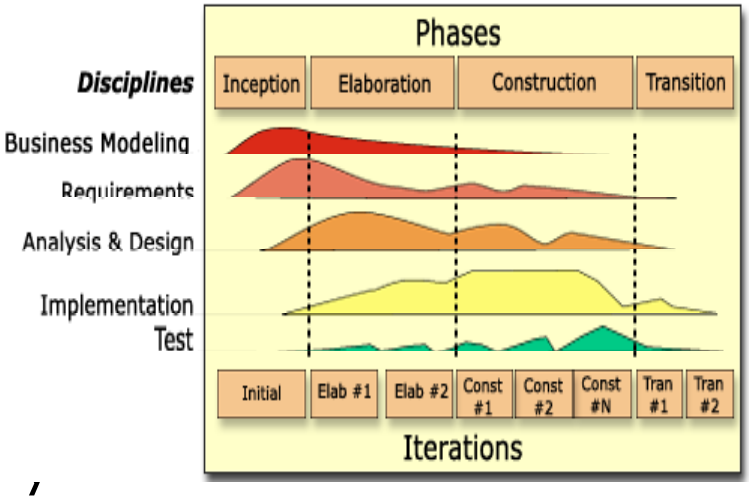
# PU – DISEÑO

| ***MODELO DE ANALISIS*** | ***MODELO DE DISEÑO*** |
|---|---|
| Modelo Conceptual, abstracción del sistema | Es un Modelo base para la implementación |
| Diseño Genérico | Diseño Específico |
| Clases estereotipadas: límite, control y entidad, sirven para identificar responsabilidades | Responsabilidades traducidas a métodos Pueden depender del lenguaje de programación |
| Menos formal | Mas Formal |
| Se busca una arquitectura estable | Se consigue una arquitectura estable |

# El Diseño en el Ciclo de Vida del Software

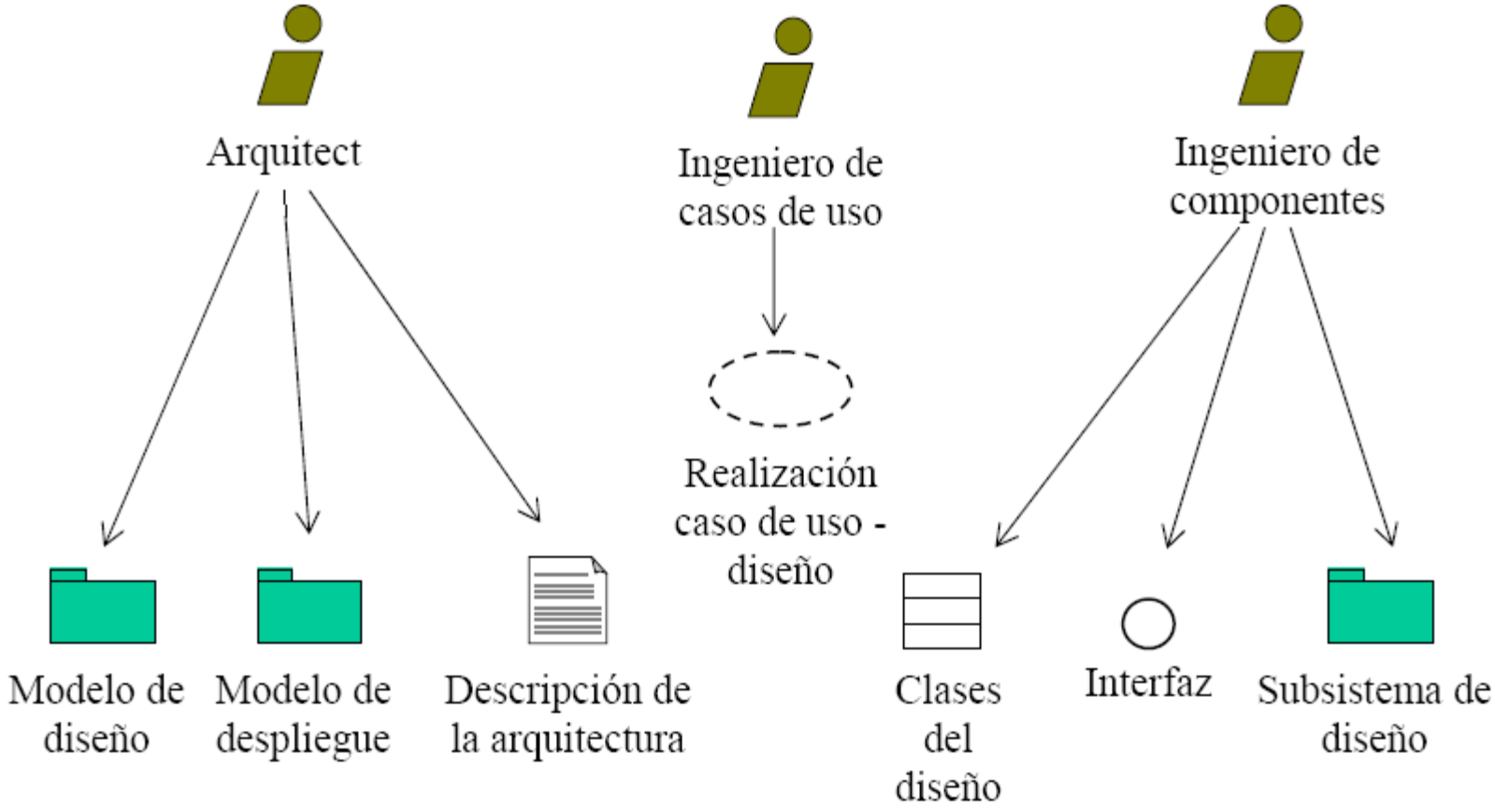
- La entrada fundamental al diseño es el modelo de análisis.
- Mayor comprensión de los Requerimientos No Funcionales: lenguaje de programación, SO, componentes de reuso, distribución y concurrencia, tecnologías de interface-usuario.

- Focalizado sobre el final de la fase de elaboración y en el inicio de la fase de construcción.




- Logra una arquitectura estable , es la base del modelo de implementación.
- Durante la fase de construcción, cuando la arquitectura es estable y los requerimientos son bien entendidos, el foco principal estará en la implementación.

# Artefactos y Workers - DISEÑO



# WORKERS - DISEÑO

- **ARQUITECTO**, responsable de la integridad del modelo de diseño y del modelo de despliegue, asegurando que dichos modelos son correctos y consistentes.

- **INGENIERO DE CASO DE USO**, responsable de la integridad de las realizaciones de caso de uso, asegurando que cumplen con los requerimientos.

- **INGENIERO DE COMPONENTES**, define y mantiene las operaciones, métodos, atributos, relaciones y requerimientos de implementación de las clases de diseño, las interfaces y los subsistemas de diseño.

# ARTEFACTOS - DISEÑO

- **Modelo de Diseño**, representa una abstracción del modelo de implementación del sistema.

  Los CU son realizados por las clases de diseño y sus objetos.

- **Clase de Diseño**, generalmente se especifican en el lenguaje de programación elegido.

  Los métodos y atributos deben quedar completamente definidos (nombre, tipo, visibilidad, parámetros, etc).

- **Realización de Caso de Uso-Diseño**, describe la manera en que cada Caso de Uso es realizado y ejecutado en términos de clases y sus objetos.

  Se modelan con Diag. Clases, Diag. Secuencia, Flujo de Eventos.

- **Subsistemas de Diseño**, organiza los artefactos del modelo de diseño en piezas más manejables.

  Un subsistema puede consistir de clases de diseño, realizaciones de casos de uso, interfaces o subsistemas.

# ARTEFACTOS - DISEÑO

- **Descripción de la arquitectura**, contiene una vista arquitectónica del modelo de diseño.

- **Modelo de Despliegue,** describe la distribución física del sistema en términos de cómo se distribuyen funcionalmente los nodos computacionales.

# ACTIVIDADES - DISEÑO

- **Diseño de la Arquitectura,** esbozar los modelos de diseño y despliegue identificando nodos y configuraciones de redes (client/server), subsistemas y sus interfaces, persistencia, distribución, seguridad.

- **Diseño de cada Caso de Uso**
  - Identificar las clases de diseño (Diagrama de Clases)
  - Describir interacción entre los objetos (Diag. de Secuencia)
  - Identificar los subsistemas participantes e interfaces.
  - Describir interacción entre subsistemas (Diag. de Secuencia)
  - Capturar requerimientos de implementación.

# ACTIVIDADES - DISEÑO

- **Diseño de cada Clase**
  - Identificar y describir Métodos
  - Identificar atributos y especificar sus tipos
  - Identificar relaciones entre clases
  - Describir Estados (Diag. de Estados, Diag. de Actividad)

- **Diseño de Subsistemas**, organizar los artefactos del diseño en piezas mas manejables, garantizando
  - que los elementos de cada subsistema son cohesivos
  - el menor acoplamiento posible entre subsistemas.

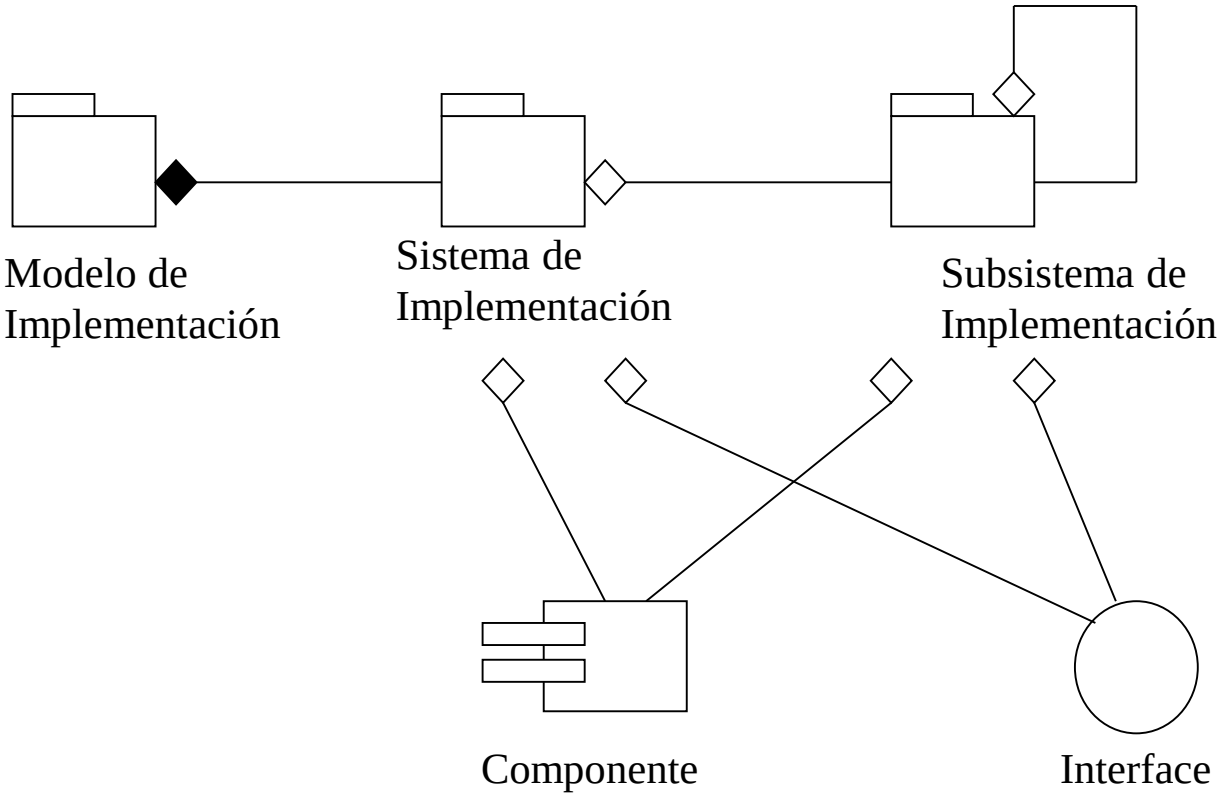
# Proceso Unificado

## Etapa: IMPLEMENTACION

## MODELO DE IMPLEMENTACION

# PU – IMPLEMENTACION

El Modelo de Implementación es una jerarquía de subsistemas de implementación que contienen los componentes e interfaces del sistema.



# Modelo de Implementación

El objetivo principal de este modelo es implementar el sistema, es decir, realizar los **casos de uso del diseño (vista lógica) a componentes del sistema (vista física)**

## ARTEFACTOS

- SUBSISTEMA DE IMPLEMENTACION
- COMPONENTE
- INTERFACES

## WORKERS

- Arquitecto
- Ingeniero de Componentes
- Integrador de Sistemas

# WORKERS y Flujo de Trabajo

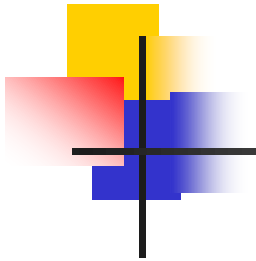
| **Arquitecto** | **Integrador del Sistema** | **Ingeniero de Componentes** |
|---|---|---|
| Implementar la arquitectura<br>---------- | ----------<br>Integrar Sistemas<br>---------- | ----------<br>✓ Implementar cada clase<br>✓ Implementar cada subsistema<br>----------<br>Realizar prueba de unidad |

# ACTIVIDADES - IMPLEMENTACION

- IMPLEMENTAR LA ARQUITECTURA:
  - Identificar los componentes ejecutables que resultan más significativos arquitectónicamente.
  - Asignar los componentes a los nodos físicos.
- INTEGRAR EL SISTEMA
  - Planificar la construcción, añadir funcionalidad a la dicha construcción.
  - Recopilar las versiones correctas de los subsistemas de implementación y de los componentes.
- IMPLEMENTAR UN SUBSISTEMA
  - Los requisitos a ser implementados en la construcción actual y los que afectan al subsistema, son correctamente implementados por componentes dentro del subsistema.

# ACTIVIDADES - IMPLEMENTACION

- IMPLEMENTAR UNA CLASE DE DISEÑO
  - Se implementa una clase de diseño y se obtiene un componente.
- REALIZAR PRUEBA DE UNIDAD
  - Probar los componentes implementados como unidades individuales.
    - Prueba de Especificación o de Caja Negra, verifica el comportamiento de una unidad observable externamente.
    - Prueba de Estructura o de Caja Blanca, verifica la implementación interna de una unidad.



# Proceso Unificado

## Etapa: PRUEBA

## MODELO DE PRUEBAS

# PU – PRUEBA

Las pruebas se centran en las fases de elaboración, cuando se prueba la línea base ejecutable de la arquitectura, y de construcción, cuando gran parte está implementado. Durante la fase de transición se corrigen los defectos y se realizan las pruebas de regresión.

**ARTEFACTOS**

- Modelo de Pruebas
- Caso de Prueba
- Procedimiento de Prueba
- Componente de Prueba
- Plan de Prueba
- Defecto
- Evaluación de Prueba

**WORKERS**

- Diseñador de Pruebas
- Ingeniero de Componentes
- Ingeniero de Pruebas de Integración
- Ingeniero de Pruebas de Sistema

# ARTEFACTOS – PRUEBA

- MODELO DE PRUEBAS

  Describe cómo se prueban los componentes ejecutables en el modelo de implementación con pruebas de integración y pruebas de sistema.

- CASO DE PRUEBA

  Especifican qué probar en el sistema. Caja Blanca: prueba la interacción interna entre los componentes del sistema, y Caja Negra: prueba el comportamiento observable externamente del sistema.

- PROCEDIMIENTO DE PRUEBA

  Especifican cómo realizar los casos de prueba.

# ARTEFACTOS – PRUEBA

- COMPONENTE DE PRUEBA

  Automatizan los procedimientos de prueba.

- PLAN DE PRUEBA

  Describe estrategias, recursos y planificación de la prueba. Define el tipo de pruebas y sus objetivos, el nivel de cobertura y el porcentaje que debería obtenerse con un resultado específico.

- DEFECTO

  Anomalía del sistema.

- EVALUACION DE PRUEBA

  Evalúa los resultados de los esfuerzos de la prueba.

# Workers y Flujo de Trabajo – PRUEBA

| Ingeniero de Pruebas | Ingeniero De Pruebas de Integración | Ingeniero de Pruebas de Sistema | Ingeniero de Componentes |
|---|---|---|---|
| Planificar Prueba<br>---<br>Diseñar Prueba<br><br>---<br>Evaluar Prueba | ---<br>Realizar prueba de integración | ---<br>Realizar prueba de sistema | ---<br>Implementar Pruebas |

# ACTIVIDADES - PRUEBA

- Planificar Prueba: planificar los esfuerzos de prueba en una iteración:
  - Describir una estrategia de prueba: que tipo de pruebas ejecutar, como y cuando y cómo determinar si el esfuerzo tiene éxito.
  - Estimar los requisitos (recursos humanos y de sistema).
  - Planificar el esfuerzo de prueba.
- Diseñar Prueba: caso de prueba y procedimiento de prueba.
  - Identificar y describir los casos de prueba.
  - Identificar los procedimientos de prueba especificando cómo realizar los casos de prueba.

# ACTIVIDADES - PRUEBA

- Implementar Prueba: automatizar procedimientos de prueba creando componentes de prueba.

- Realizar Pruebas de Integración: casos de prueba para verificar que los componentes interaccionan entre si de la forma apropiada después de haber sido integrados en una construcción. Pasos:

  1. Realizar pruebas de integración relevantes, manuales o ejecutar algún componente de prueba.

  2. Comparar los resultados obtenidos con los esperados e investigar los que no coinciden.

  3. Informar defectos a los Ing. de componentes.

  4. Informar defectos a los diseñadores de pruebas.

# ACTIVIDADES - PRUEBA

- Realizar Pruebas de Sistema:
  - Probar que el sistema funciona correctamente como un todo.
  - Si los CU satisfacen los requerimientos de los actores.
  - Probar con diferentes configuraciones (SW y HW), diferentes cargas al sistema, diferentes número de actores y distintos tamaños de BD.
- Evaluar las Pruebas: evaluar los esfuerzos de la prueba en una iteración. Se usan distintas métricas que permiten determinar el nivel de calidad del software y qué cantidad de pruebas es necesario realizar.

# Bibliografía Consultada


- Jacobson Ivar, Booch Grady, Rumbaugh James. The Unified Software Development Process . Addison Wesley. 1999.

- http://www.rational.com/rup