

Metodología de Sistemas I

Gustavo Julián Rivas



Proceso Unificado: guiado por los casos de uso

¿Qué es el Proceso Unificado?

- Es una metodología de desarrollo de software iterativa e incremental.
- Propuesto por Rational Software y estandarizado por sus autores (Jacobson, Booch, Rumbaugh).
- Integra las mejores prácticas del desarrollo de software en una estructura adaptable.
- Usa UML como lenguaje estándar de modelado.

Fases del Proceso Unificado

Inicio: definición del alcance, objetivos de negocio y riesgos principales.

Elaboración: análisis del dominio, arquitectura base y casos de uso clave.

Construcción: desarrollo completo del sistema, iteración tras iteración.

Transición: puesta en producción y ajustes finales.



Cada fase incluye iteraciones internas con entregas parciales.

Principios clave del Proceso Unificado

- Desarrollo iterativo e incremental.
- Gestión de requerimientos continua.
- Arquitectura basada en componentes.
- Verificación temprana y continua.
- Control de cambios e integraciones frecuentes.

👉 Todos los procesos giran en torno a los **casos de uso**.

Captura de Requisitos en el Proceso Unificado

Importancia de los requisitos

- Un requisito es una necesidad, expectativa o condición que debe cumplir un sistema.
- Determina **qué** debe hacer el software, no necesariamente **cómo**.
- Una mala captura de requisitos genera fallas, retrabajo y pérdida de valor.

Tipos de requisitos

Requisitos funcionales: describen funcionalidades del sistema (ej. registrar usuario).

Requisitos no funcionales: calidad del servicio (rendimiento, seguridad, usabilidad).

Restricciones: condiciones impuestas (tecnología, compatibilidad, normas legales).

Captura de requisitos centrada en el usuario

- El Proceso Unificado promueve la captura de requisitos desde la perspectiva del usuario.
 - Se usan **casos de uso** como mecanismo principal para entender las interacciones.
- 🎯 Objetivo: garantizar que el sistema se alinee con las necesidades reales de negocio.

¿Por qué usar casos de uso?

- Porque representan escenarios de interacción entre el sistema y los actores.
 - Permiten capturar comportamientos observables del sistema.
 - Fomentan la colaboración entre usuarios, analistas y desarrolladores.
- ✓ Son una herramienta **central** para alinear visión técnica y negocio.

¿Qué es un Caso de Uso?

Es una **narrativa o secuencia de acciones** que describe una interacción entre un actor externo (usuario u otro sistema) y el sistema, con un objetivo específico.

 Ejemplo: “Emitir una factura”, “Consultar saldo”, “Registrar compra”.

Elementos principales de un caso de uso

- **Nombre:** describe claramente la intención (verbo + objeto).
- **Actor(es):** quien inicia la interacción.
- **Escenario principal:** flujo normal de acciones.
- **Escenarios alternativos:** excepciones, condiciones especiales.
- **Precondiciones:** lo que debe cumplirse antes.
- **Postcondiciones:** lo que debe cumplirse después.

Actor del caso de uso

- Representa un **rol externo** al sistema.
- Puede ser una persona, otro sistema o un dispositivo.

Ejemplos:



Cliente



Sistema de pagos



Aplicación móvil externa

Nivel de detalle de un caso de uso

- Casos de uso pueden escribirse con distintos niveles de detalle:
 - **Breve:** una o dos líneas.
 - **Casos principales:** narrativa estructurada (plantilla estándar).
 - **Extendidos:** incluyen reglas de negocio, diagramas, validaciones.

 **El nivel depende del objetivo y la audiencia.**

¿Qué NO es un caso de uso?

- No es una especificación técnica.
- No es un diagrama de clases.
- No describe interfaces gráficas.
- No detalla el diseño interno del sistema.



Es una **visión externa y funcional** del sistema.

Elementos de un Caso de Uso

Plantilla típica para describir un caso de uso

Campo

Nombre

Actores

Precondiciones

Flujo principal

Flujos alternativos

Postcondiciones

Descripción

“Realizar compra”

Cliente

Cliente registrado

Paso 1, Paso 2, ...

Error de pago, cliente sin stock

Pedido registrado

Flujo principal de eventos

- Describe la secuencia **normal** de pasos desde el inicio hasta el final exitoso.
- Cada paso representa una acción del actor o del sistema.
- Debe estar enfocado en el **valor que recibe el actor**.

 Ejemplo:

1. Cliente selecciona productos.
2. Cliente confirma carrito.
3. Sistema solicita pago.
4. Cliente ingresa datos de tarjeta.
5. Sistema aprueba la compra.

Flujos alternativos y de excepción

- **Flujos alternativos:** situaciones donde el flujo toma un camino distinto pero válido.
- **Flujos de excepción:** errores o condiciones que interrumpen el flujo.

 Ejemplo:

Flujo principal: “Cliente realiza pago”.

Flujo alternativo: “Cliente elige pagar con crédito en vez de débito”.

Flujo de excepción: “Transacción rechazada por el banco”.

Ejemplo (resumen): “Registrar Usuario”

- **Nombre:** Registrar Usuario
- **Actor principal:** Visitante
- **Precondición:** No estar registrado
- **Postcondición:** Usuario registrado con cuenta activa
- **Flujo principal:**
 1. El visitante accede al formulario
 2. Ingresa sus datos
 3. El sistema valida la información
 4. El sistema crea el usuario y muestra mensaje de éxito

Caso de uso: Registrar Usuario


Campo	Descripción
Actores	Visitante
Precondiciones	No debe estar autenticado
Flujo Principal	<ol style="list-style-type: none">1. Ingresa al formulario.2. Completa los campos requeridos.3. Presiona "Registrar".4. Sistema valida campos.5. El sistema guarda el nuevo usuario.6. Se muestra mensaje de éxito.
Flujos Alternativos	A1: Usuario ya registrado A2: Email inválido
Postcondiciones	Usuario con cuenta activa en el sistema

Ejemplo 2 (resumen): “Realizar Compra”

- **Actor principal:** Cliente
- **Precondición:** Cliente autenticado con productos en el carrito
- **Postcondición:** Compra registrada y pago confirmado
- **Flujo principal:**
 1. Cliente revisa su carrito
 2. Confirma productos
 3. Selecciona método de pago
 4. Ingresar datos y confirma
 5. Sistema procesa el pago

Relación entre casos de uso y arquitectura

Los casos de uso:

- Guiarán el diseño arquitectónico
 - Ayudan a definir responsabilidades del sistema
 - Permiten identificar clases, servicios y componentes necesarios
-  Cada caso de uso se convierte en una “unidad de funcionalidad entregable”

¿Cómo identificar casos de uso?

1. Identificar los **actores**
2. Listar los **objetivos** que persiguen
3. Agrupar los escenarios similares
4. Confirmar con los usuarios

✓ Técnica: entrevistas, talleres, observación

Criterios para definir casos de uso

- Cada uno debe generar **valor observable**
- Tener un **inicio y fin claros**
- Representar una **interacción completa**
- Evitar casos de uso demasiado grandes (“épicos”) o triviales

Organización jerárquica de casos de uso

Algunos casos de uso pueden:

- Incluir otros casos (composición)
- Ser especializados (herencia)

 Ayuda a modularizar el sistema y reducir la duplicación

Casos de uso vs funcionalidades técnicas

✗ “Validar email” NO es un caso de uso

✓ “Registrar usuario” Sí lo es

🔍 Pensar siempre desde la perspectiva del actor, no del programador

Beneficios del Proceso Guiado por Casos de Uso

- Mejora la comprensión de requisitos
- Mejora la comunicación entre técnicos y no técnicos
- Facilita pruebas basadas en escenarios
- Permite priorizar funcionalidad por iteraciones

Casos de uso en el ciclo de vida del software

- **Inicio:** ayudan a definir alcance
- **Elaboración:** permiten estimar esfuerzo
- **Construcción:** guían el diseño y desarrollo
- **Transición:** son base para pruebas de aceptación

Casos de uso y testing

- Se convierten fácilmente en **escenarios de prueba funcional**
- Aportan trazabilidad completa desde requerimiento hasta validación
- Facilitan automatización de pruebas por comportamiento (BDD)

Casos de uso y gestión de cambios

- Los cambios se rastrean por impacto en uno o más casos
- Permite evaluar fácilmente qué funcionalidades están involucrada
- Mejora la gestión del riesgo

Buenas prácticas

- ✓ Usar lenguaje claro y centrado en el usuario
- ✓ Validar los casos con los usuarios finales
- ✓ Documentar flujos alternativos y errores
- ✓ Reutilizar patrones comunes (login, registro, etc.)

Errores comunes

- ✗ Casos mal nombrados (“Pantalla de login”)
- ✗ Confundir tareas técnicas con funcionalidades
- ✗ No detallar flujos de excepción
- ✗ Redactar desde la lógica del sistema, no del usuario