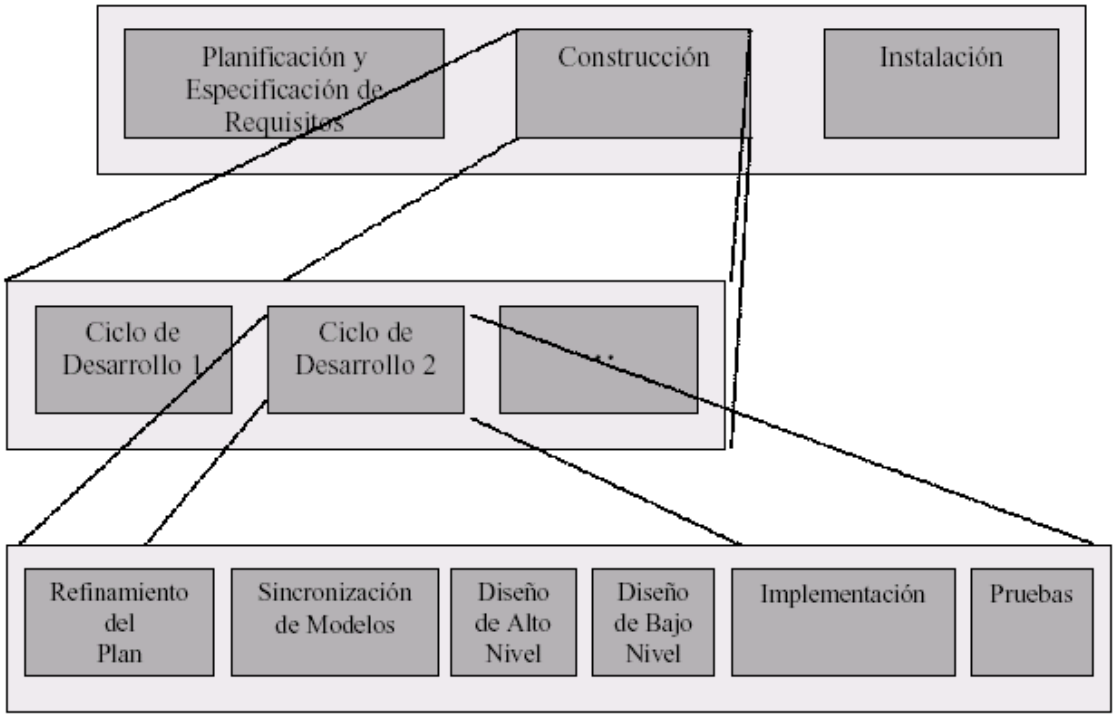
Desarrollo Orientado a Objetos con UML
C.E.C.yT. "Juan de Dios Bátíz Paredes" – IPN

Resumido y Reorganizado por Lic. Guillermo Cherencio

INDICE

La metodología propuesta por Craig Larman[1] se divide en 3 etapas: **1) Planificación y Especificación de Requisitos, 2) Construcción 3) Instalación**. La etapa 2) es evolutiva, cíclica, interativa, incremental, en cada iteración (llamada ciclo de desarrollo) se toma un subconjunto de requisitos para su diseño e implementación. Todo el proceso puede resumirse en:



1. Planificación y Especificación de Requisitos:

Planificación, definición de requisitos, construcción de prototipos. Fase independiente del paradigma a utilizar.

1.1 Actividades:

1. Definir el Plan-Borrador.
2. Crear el Informe de Investigación Preliminar.
3. Definir los Requisitos.
4. Registrar Términos en el Glosario. (continuado en fases posteriores)
5. Implementar un Prototipo. (opcional)
6. Definir Casos de Uso (de alto nivel y esenciales).
7. Definir el Modelo Conceptual-Borrador. (puede retrasarse hasta una fase posterior)
8. Definir la Arquitectura del Sistema-Borrador. (puede retrasarse hasta una fase posterior)

[1] UML y Patrones. C. Larman. Prentice Hall, 1999

9. Refinar el Plan.

1.2 Requisitos:

Identificar qué es lo que realmente se necesita. Realizar documento de especificación de requisitos con los siguientes puntos:

- Propósito.
- Ámbito del Sistema, Usuarios.
- Funciones del Sistema.
- Atributos del Sistema.

1.3 Casos de Uso

Describe la secuencia de eventos de un actor (un agente externo) que usa un sistema para completar un proceso. Describirlo a través de un texto, en forma abstracta (no es necesario en esta etapa realizar los diagramas de casos de uso). Dividir a los casos de uso en las siguientes categorías:

1.3.1 Casos de Uso de alto nivel, ejemplo:

Caso de Uso: Realizar Reintegro
Actores: Cliente
Tipo: primario
Descripción: Un Cliente llega al cajero automático, introduce la tarjeta, se identifica y solicita realizar una operación de reintegro por una cantidad específica. El cajero le da el dinero solicitado tras comprobar que la operación puede realizarse. El Cliente toma el dinero y la tarjeta y se va.

Es un caso de uso descrito a alto nivel, la descripción es muy general, normalmente se condensa en dos o tres frases. Es útil para comprender el ámbito y el grado de complejidad del sistema.

1.3.2 Casos de Uso Expandidos, tomar los casos de uso más importantes y expresarlos de forma más detallada, ejemplo:

Caso de Uso: Realizar Reintegro
Actores: Cliente (iniciador)
Propósito: Realizar una operación de reintegro de una cuenta del banco.
Visión General: Un Cliente llega al cajero automático, introduce la tarjeta, se identifica y solicita realizar una operación de reintegro por una cantidad específica. El cajero le da el dinero solicitado tras comprobar que la operación puede realizarse. El Cliente toma el dinero y la tarjeta y se va.
Tipo: primario y esencial.
Referencias: Funciones: R1.3, R1.7
Curso Típico de Eventos:

Acción del Actor	Respuesta del Sistema
1. Este caso de uso empieza cuando un Cliente introduce una tarjeta en el cajero.	
	2. Pide la clave de identificación.

3. Introduce la clave.	
	4. Presenta las opciones de operaciones disponibles.
5.Selecciona la operación de Reintegro.	
	6. Pide la cantidad a retirar.
7. Introduce la cantidad requerida.	
	8. Procesa la petición y, eventualmente, da el dinero solicitado. Devuelve la tarjeta y genera un recibo.
9. Recoge el dinero, el recibo y la tarjeta, y se va.	

Cursos Alternativos:
Línea 4: La clave es incorrecta. Se indica el error y se cancela la operación.
Línea 8: La cantidad solicitada supera el saldo. Se indica el error y se cancela la operación.

1.3.3 Identificacion de Casos de Uso

Usar *brainstorming* entre los miembros del equipo. Para la identificación inicial se usan 2 métodos:
a) basado en actores: identificar los actores relacionados con el sistema. Para cada actor, identificar los procesos que inicia o participa.
b) basado en eventos: identificar los eventos externos a los que el sistema tiene que responder. Relacionar los eventos con actores y casos de uso.

1.3.4 Identificación de los límites del sistema

El límite del sistema da significado a los casos de uso, permite diferenciar entre interno y externo al sistema.

1.3.5 Tipos de Caso de Uso

a) según importancia:
primarios: procesos principales, más comunes, ej. realizar reintegro
secundarios: de uso infrecuente, ej. añadir nueva operación
opcionales: procesos que pueden o no ser abordados en el presente proyecto (limite)
b) según el grado de compromiso con el diseño:
las descripciones de casos de uso anteriores son a nivel abstracto, sin compromiso con la solución, independiente de la tecnología e implementación, estos son los casos de uso **esenciales**. Un caso de uso **real** describe concretamente el proceso en términos del diseño real, de la solución a llevarse a cabo, se ajusta a una interfaz determinada y se baja a detalles tales como pantallas y objetos.

1.3.6 Consejos relativos a Casos de Uso

a) nombre: verbo, enfatizar que es un proceso, ej. comprar artículos

b) alternativas equiprobables: indicar en el apartado Cursos Alternativos y Curso Tipico de Eventos en distintas secciones adicionales, ejemplo:

-Curso Típico de Eventos

-Sección: Principal

Acción del Actor	Respuesta del Sistema
1. Este caso de uso empieza cuando Actor llega alsistema.	
	2. Pide la operación a realizar.
3. Escoge la operación A.	
	4. Presenta las opciones de pago.
5. Selecciona el tipo de pago: a) Si se paga al Contado, ver sección Pago al Contado b) Si se paga con Tarjeta, ver sección Pago con Tarjeta	
	6. Genera Recibo.
7. Recoge el recibo y se va.	

Cursos Alternativos:

Lineas 3 y 5: Selecciona Cancelar. Se cancela la operación

-Sección Pago al Contado

Acción del Actor	Respuesta del Sistema
1. Mete los billetes correspondientes.	
	2. Toma los billetes y sigue pidiendo dinero hasta que la cantidad está satisfecha.
.	3. Devuelve el cambio

... y así sucesivamente para la sección Pago con Tarjeta

1.4 Construcción del Modelo de Casos de Uso

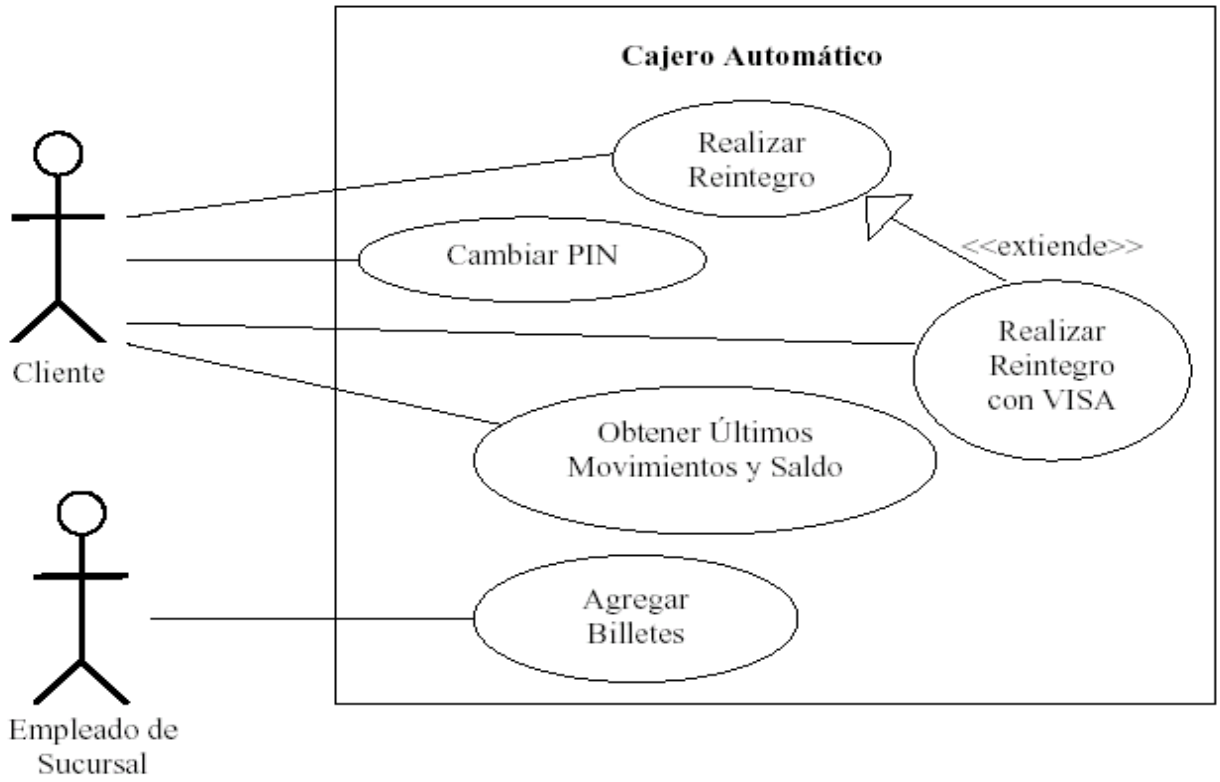
Seguir los siguientes pasos:

1. listar las funciones del sistema, definir los límites del sistema, identificar actores y casos de uso.
2. escribir todos los casos de uso en el formato de alto nivel. Categorizarlos como primarios, secundarios u opcionales.
3. dibujar Diagramas de Casos de Uso.
4. relacionar los casos de uso e ilustrar las relaciones en el Diagrama de Casos de Uso (<<extiende>> y <<usa>>).
5. los casos de uso más críticos, importantes y que conllevan un mayor riesgo, se describen en el formato expandido esencial. Se deja la definición en formato expandido esencial del resto de casos de uso para cuando sean tratados en posteriores ciclos de desarrollo, evitar tratar toda la complejidad del problema de una sola vez.
6. se crean casos de uso reales sólo cuando:

a) las descripciones más detalladas ayudan significativamente a incrementar la comprensión del problema.

b) el cliente pide que los procesos se describan de esta forma.

7. ordenar según prioridad los casos de uso



1.5 Planificación de Casos de Uso según ciclos de desarrollo

¿Qué partes se van a abordar en cada ciclo de desarrollo? Se van a tomar uno o mas casos de uso por ciclo, comenzando por los más importantes según:

a. Impacto significativo en el diseño de la arquitectura. Por ejemplo, si aporta muchas clases al modelo del dominio o requiere persistencia en los datos.
b. Se obtiene una mejor comprensión del diseño con un nivel de esfuerzo relativamente bajo.
c. Incluye funciones complejas, críticas en el tiempo o de nivel elevado de riesgo.
d. Implica un trabajo de investigación significativo, o bien el uso de una tecnología nueva o arriesgada.
e. Representa un proceso de gran importancia en la línea de negocio.
f. Supone directamente un aumento de beneficios o una disminución de costos.

1.5.1 Caso de Uso Inicialización

El caso de uso Inicialización hará todas las acciones que sean necesarias para satisfacer las necesidades de inicialización de casos de uso que se tratan en cada ciclo, de esta forma, se puede obtener una versión del sistema que pueda funcionar.

2. Construcción

La construcción del sistema se realiza en las siguientes fases:

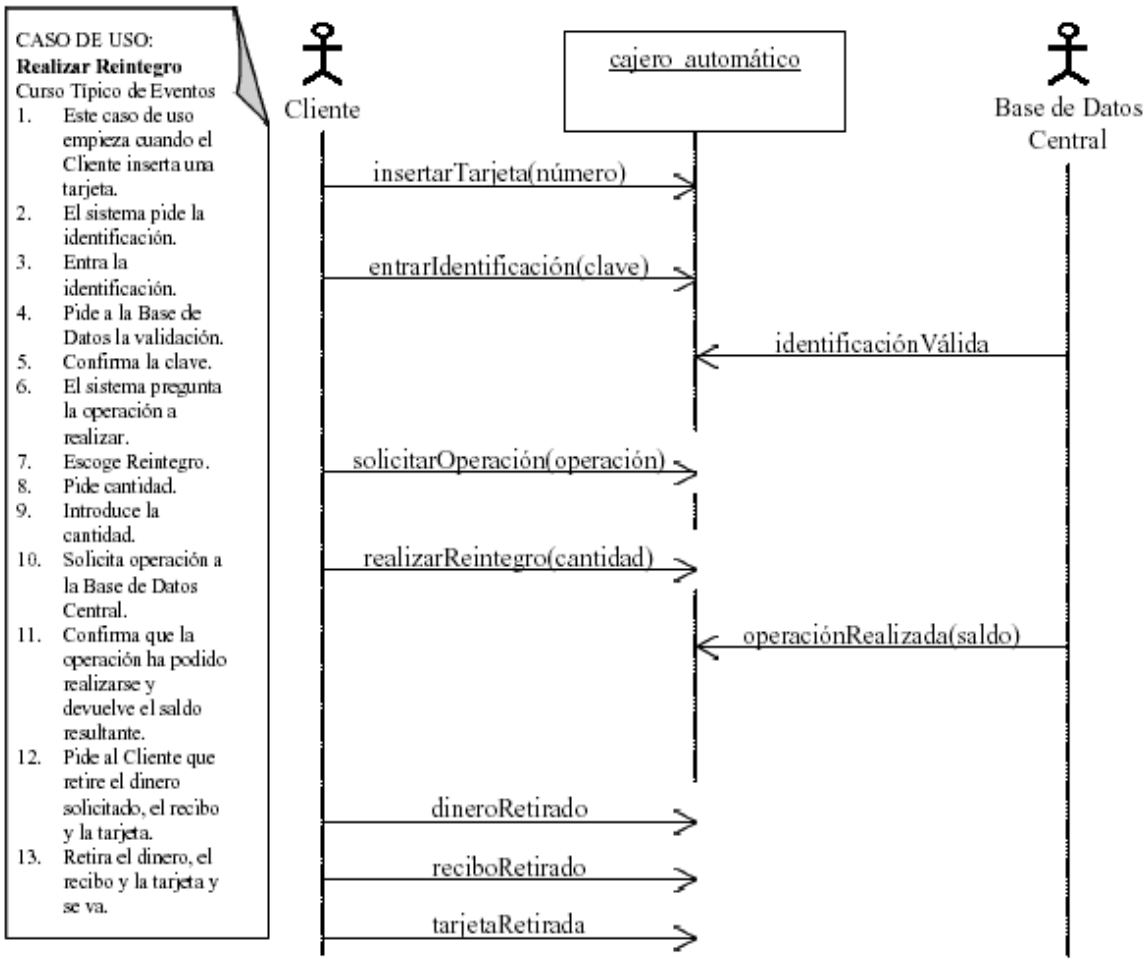
2.1 Diseño de Alto Nivel: Se aborda el problema viendo al sistema a construir como una caja negra, centrándonos en la visión que desde el exterior tienen los actores, esto es, en los casos de uso. Se analiza el problema construyendo un modelo conceptual.

2.1.1 Actividades de la fase de Diseño de Alto Nivel:

1. Definir Casos de Uso Esenciales en formato expandido. (si no están definidos)
2. Refinar los Diagramas de Casos de Uso.
3. Definir los Diagramas de Secuencia del Sistema.
4. Refinar el Modelo Conceptual.
5. Refinar el Glosario. (continuado en posteriores fases)
6. Definir Contratos de Operación.
7. Definir Diagramas de Estado. (opcional)

2.1.2 Diagramas de Secuencia del Sistema

Para cada caso de uso se van a construir una serie de Diagramas de Secuencia que muestren los eventos que llegan al sistema (y no los que salen) para cada escenario del caso de uso. En la interacción de actores con el sistema se generan eventos, solicitando operaciones al sistema. Una instancia de un caso de uso se denomina escenario y muestra la ejecución real del caso de uso, con las posibles bifurcaciones y alternativas resueltas en forma particular:



Los eventos del sistema deberían expresarse en base a la noción de operación que representan, en vez de en base a la interfaz particular (ej. se prefiere “finOperacion” a “presionadaTeclaEnter”)

Con el diagrama de secuencia identificamos las operaciones del sistema y para cada una de ellas definiremos un contrato (qué se espera de dicha operación). El contrato se basa en los cambios que sufren los elementos del Modelo Conceptual en cada operación.

2.1.3 Modelo Conceptual

El modelo conceptual representa conceptos del mundo real a través de un diagrama de estructura estática UML que se llama Modelo Conceptual. Su objetivo es aumentar la comprensión del problema, es preferible crear un modelo con muchos conceptos a quedarse corto y olvidar algunos conceptos importantes.

2.1.3.1 Identificación de conceptos

Basarse en los requisitos, en la descripciones de casos de uso y en el conocimiento gral. del dominio del problema, ejemplos:

Tipo de Concepto	Ejemplo
Objetos físicos o tangibles	Avión Terminal_de_Caja
Especificaciones, diseños o descripciones de cosas	Especificacion_de_Producto Descripcion_de_Vuelo
Lugares	Supermercado Aeropuerto
Transacciones	Venta, Pago Reserva
Líneas de una transacción	Artículo_de_Venta
Roles de una persona	Cajero Piloto
Contenedores de otras cosas	Supermercado, Cesta Avión
Cosas en un contenedor	Artículo Pasajero
Otros ordenadores o sistemas electromecánicos externos a nuestro sistema	Sistema_de_Autorización_de_Tarjetas_de Crédito Sistema_Controlador_de_Tráfico_Aéreo
Conceptos abstractos	Hambre
Organizaciones	Departamento_de_Ventas Compañía_Aérea_Toto
Eventos	Venta, Robo, Reunión Vuelo, Accidente, Aterrizaje
Reglas y políticas	Política_de_Devoluciones Política_de_Cancelaciones
Catálogos	Catálogo_de_Productos Catálogo_de_Piezas
Archivos financieros, de trabajo, de contratos, de asuntos legales.	Recibo, Contrato_de_Empleo Registro_de_Revisiones
Instrumentos y servicios financieros	Línea_de_Crédito Stock
Manuales, libros	Manual_del_Empleado Manual_de_Reparaciones

2.1.3.2 Creación del Modelo Conceptual

Seguir los siguientes pasos:

1. Hacer una lista de conceptos candidatos usando la Tabla 1 y la búsqueda de sustantivos relacionados con los requisitos en consideración en este ciclo.
2. Representarlos en un diagrama.
3. Añadir las asociaciones necesarias para ilustrar las relaciones entre conceptos que es necesario conocer.

4. Añadir los atributos necesarios para contener toda la información que se necesite conocer de cada concepto.

2.1.3.3 Identificación de las Asociaciones

Incluir al modelo conceptual las siguientes asociaciones:
a) Asociaciones para las que el conocimiento de la relación necesita mantenerse por un cierto período de tiempo (asociaciones “necesita-conocer”).
b) Asociaciones derivadas de la Lista de Asociaciones Típicas:

Categoría	Ejemplos
A es una parte física de B	Ala - Avion
A es una parte lógica de B	Articulo_en_Venta - Venta
A esta físicamente contenido en B	Articulo - Estanteria Pasajero - Avion
A esta lógicamente contenido en B	Descripcion_de_Articulo - Catalogo
A es una descripción de B	Descripcion_de_Articulo - Articulo
A es un elemento en una transacción o un informe B	Trabajo_de_Reparacion - Registro_de_Reparaciones
A es registrado/archivado/capturado en B	Venta - Terminal_de_Caja
A es un miembro de B	Cajero - Supermercado Piloto - Compania_Aerea
A es una subunidad organizativa de B	Seccion - Supermercado Mantenimiento - Compania_Aerea
A usa o gestiona B	Cajero - Terminal_de_Caja Piloto - Avion
A comunica con B	Cliente - Cajero Empleado_de_Agencia_de_Viajes - Pasajero
A esta relacionado con una transacción B	Cliente - Pago Pasajero - Billete
A es una transacción relacionada con otra transacción B	Pago - Venta Reserva - Cancelacion
A esta junto a B	Ciudad - Ciudad
A posee B	Supermercado - Terminal_de_Caja Compania_Aerea - Avion

2.1.3.4 Identificación de Atributos

Incluir los atributos necesarios para satisfacer las necesidades de información de los casos de uso. Los atributos deben tomar valores simples (número, texto, etc).

2.1.4 Glosario

Poner una descripción textual de todo concepto de cualquier modelo para eliminar toda posible ambigüedad, ejemplo:

Término	Categoría	Descripción
Realizar Reintegro	caso de uso	Descripción del proceso por el que un cliente realiza un reintegro en un cajero automático
Banco	concepto	Entidad que ofrece servicios financieros a sus clientes
...	...	...

2.1.5 Contratos de Operaciones

Se describe mediante contratos el comportamiento esperado del sistema en cada operación. Contrato=documento que describe qué es lo que se espera de una operación, formato declarativo, enfatizando en el *qué* más que en el *cómo*. Se expresan en forma de pre y post condiciones en torno a cambios de estado. Se puede escribir un contrato para un método individual de una clase o para una operación completa del sistema (aquí se toma este ultimo):

Contrato

Nombre: InsertarTarjeta (número_tarjeta: número)
Responsabilidades: Comenzar una sesión con el sistema para realizar una operación. Presentar las opciones disponibles.
Tipo: Sistema
Referencias a Funciones del Sistema: R1.2, R1.6, R1.7
Cruzadas:
Casos de Uso: Reintegro
Notas:
Excepciones: Si la tarjeta es ilegible, indicar que ha habido un error.
Salida:
Pre-condiciones: No hay una sesión activa.
Post-condiciones:

-Una nueva Sesión se ha creado. (creación de instancia).
-La Sesión se ha asociado con Cajero. (asociación formada).

Descripción de los items de un contrato:

Item	Descripción
Nombre:	Nombre de la operación y parámetros.
Responsabilidades:	Una descripción informal de las responsabilidades que la operación debe desempeñar.
Tipo:	Nombre del tipo (clase software, sistema).
Referencias Cruzadas:	Números de referencia en los requisitos de funciones del sistema, casos de uso, etc.
Notas:	Comentarios de diseños, algoritmos, etc.
Excepciones:	Casos excepcionales.
Salida:	Salidas que no corresponden a la interfaz de usuario, como mensajes o registros que se envían fuera del sistema.
Pre-condiciones:	Asunciones acerca del estado del sistema antes de ejecutar la operación.
Post-condiciones:	El estado del sistema después de completar la operación.

2.1.5.1 Construcción de un Contrato

1. Identificar las operaciones del sistema a partir de los Diagramas de Secuencia del Sistema.
2. Para cada operación del sistema construir un contrato.
3. Empezar escribiendo el apartado de Responsabilidades, describiendo informalmente el propósito de la operación.
4. A continuación rellenar el apartado de Post-condiciones, describiendo declarativamente los cambios de estado que sufren los objetos en el Modelo Conceptual.
5. Para describir las post-condiciones, usar las siguientes categorías:
    - Creación y borrado de instancias.
    - Modificación de atributos.
    - Asociaciones formadas y retiradas.
6. Completar el resto de apartados en su caso.

2.1.5.2 Post-condiciones

Es la parte más importante de un contrato, se basa en los cambios que ha sufrido el modelo conceptual una vez que se ha realizado la operación. Usar tiempo pasado o pretérito perfecto (ej. “se ha creado una sesión”), acordarse de añadir asociaciones a los objetos creados.

2.1.6 Diagramas de Estados

Diagramas de Estados de UML aplicables al comportamiento de: una clase, un concepto o un caso de uso. En esta fase (alto nivel) sólo se harían para los casos de uso más complejos. Su utilidad reside en mostrar la secuencia permitida de eventos externos que pueden ser reconocidos y tratados por el sistema (por ej. no se puede insertar una tarjeta en un cajero automático si se está en el transcurso de una operación).

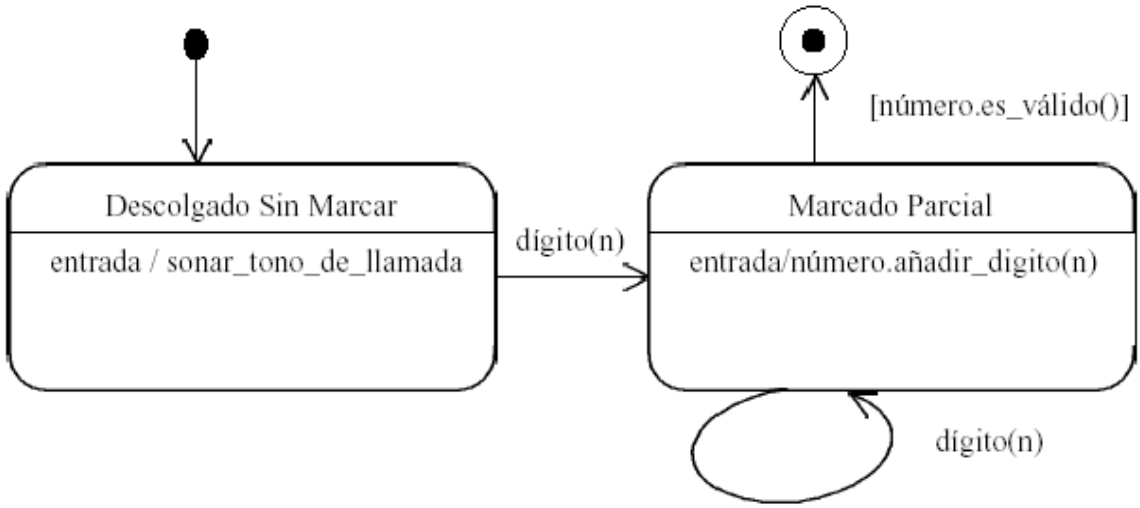


Diagrama de Estados

2.2 Diseño de Bajo Nivel: El sistema definido en la fase anterior se especifica en detalle, describiendo todas las operaciones que el sistema va a tener que realizar internamente para satisfacer lo especificado en el diseño de alto nivel.

2.2.1 Actividades

1. Definir los Casos de Uso Reales.
2. Definir Informes e Interfaz de Usuario.
3. Refinar la Arquitectura del Sistema.
4. Definir los Diagramas de Interacción.
5. Definir el Diagrama de Clases de Diseño. (en paralelo con los Diagramas de Interacción)
6. Definir el Esquema de Base de Datos.

El paso de Refinar la Arquitectura del Sistema no tiene por qué realizarse en la posición 3, puede realizarse antes o después.

2.2.2 Casos de Uso Reales

Describe el diseño real del caso de uso según una tecnología concreta de entrada y de salida y su implementación. Si el caso de uso implica una interfaz de usuario, se incluirá bocetos de las ventanas y detalles de la interacción a bajo nivel con los *widgets* de la ventana. Como alternativa, el desarrollador puede crear bocetos de la interfaz en papel, y dejar los detalles para la fase de implementación.

2.2.3 Diagramas de Colaboración

Los Diagramas de Interacción muestran el intercambio de mensajes entre instancias del modelo de clases para cumplir las post-condiciones establecidas en un contrato.

Hay dos clases de Diagramas de Interacción:

1. Diagramas de Colaboración.
2. Diagramas de Secuencia.

De entre ambos tipos se prefieren los Diagramas de Colaboración por su expresividad y por su economía espacial (una interacción compleja puede ser muy larga en un Diagrama de Secuencia).

La creación de los Diagramas de Colaboración de un sistema **es una de las actividades más importantes en el desarrollo orientado a objetos**, pues al construirlos se toman unas decisiones clave acerca del funcionamiento del futuro sistema. La creación de estos diagramas, por tanto, **debería ocupar un porcentaje significativo en el esfuerzo dedicado al proyecto entero**.

2.2.3.1 Creación de Diagramas de Colaboración

Crear un diagrama separado para cada operación del sistema en desarrollo en el ciclo de desarrollo actual.Para cada evento del sistema, hacer un diagrama con él como mensaje inicial.

-Si el diagrama se complica, dividirlo en diagramas más pequeños.

-Usando los apartados de responsabilidades y de post-condiciones del contrato de operación, y la descripción del caso de uso como punto de partida, diseñar un sistema de objetos que interaccionan para llevar a cabo las tareas requeridas.

La capacidad de realizar una buena asignación de responsabilidades a los distintos objetos es una habilidad clave, y se va adquiriendo según aumenta la experiencia en el desarrollo orientado a objetos.

Booch, Rumbaugh y Jacobson definen **responsabilidad** como **"un contrato u obligación de una clase o tipo"**[2]. Las responsabilidades están ligadas a las obligaciones de un objeto en cuanto a su comportamiento. Básicamente, estas responsabilidades son de los dos siguientes tipos:

a) Conocer:

Conocer datos privados encapsulados.
Conocer los objetos relacionados.
Conocer las cosas que puede calcular o derivar.

b) Hacer:

Hacer algo él mismo.
Iniciar una acción en otros objetos.
Controlar y coordinar actividades en otros objetos.

Por ejemplo, puedo decir que "un Recibo es responsable de imprimirse" (tipo hacer), o que "una Transacción es responsable de saber su fecha" (tipo conocer). Las responsabilidades de tipo "conocer" se pueden inferir normalmente del Modelo Conceptual.

Una responsabilidad no es lo mismo que un método, pero los métodos se implementan para satisfacer responsabilidades.

[2] "The UML Specification Document", G. Booch, I. Jacobson and J.Rumbaugh. Rational Software Corp., 1997

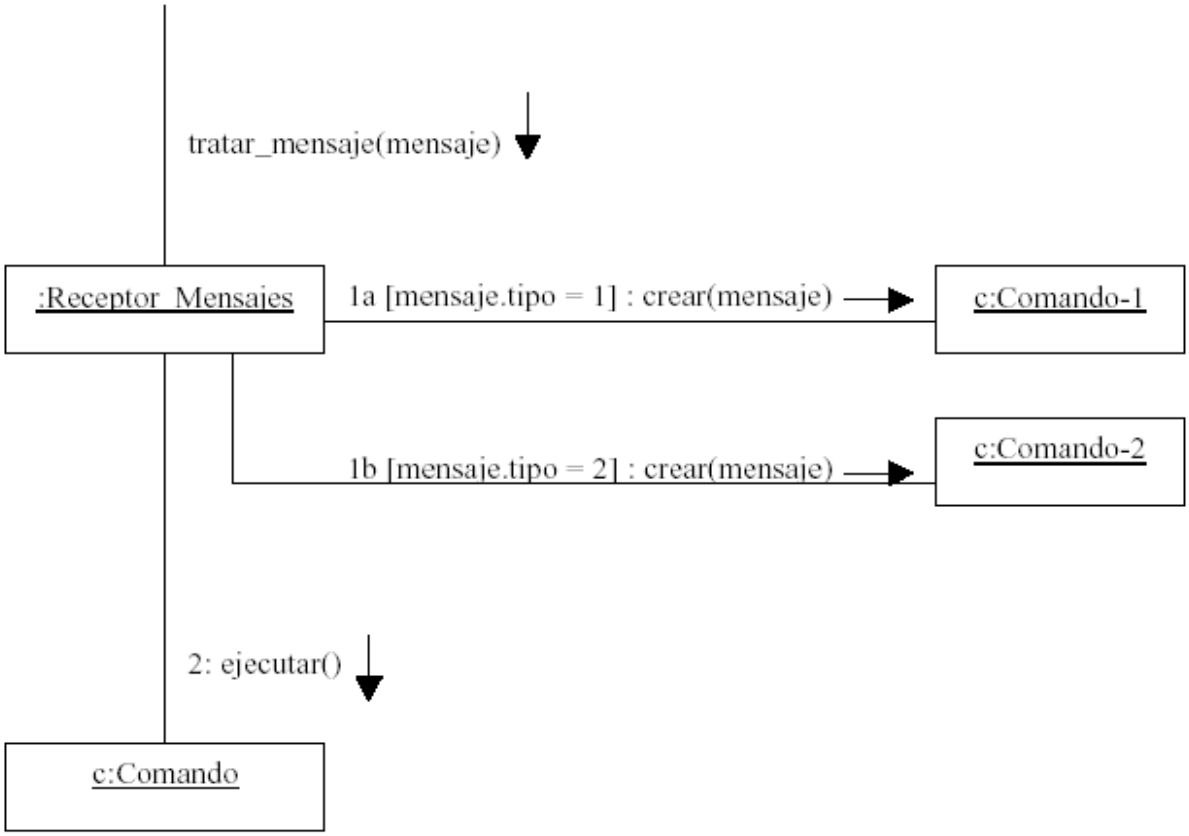


Diagrama de Colaboración

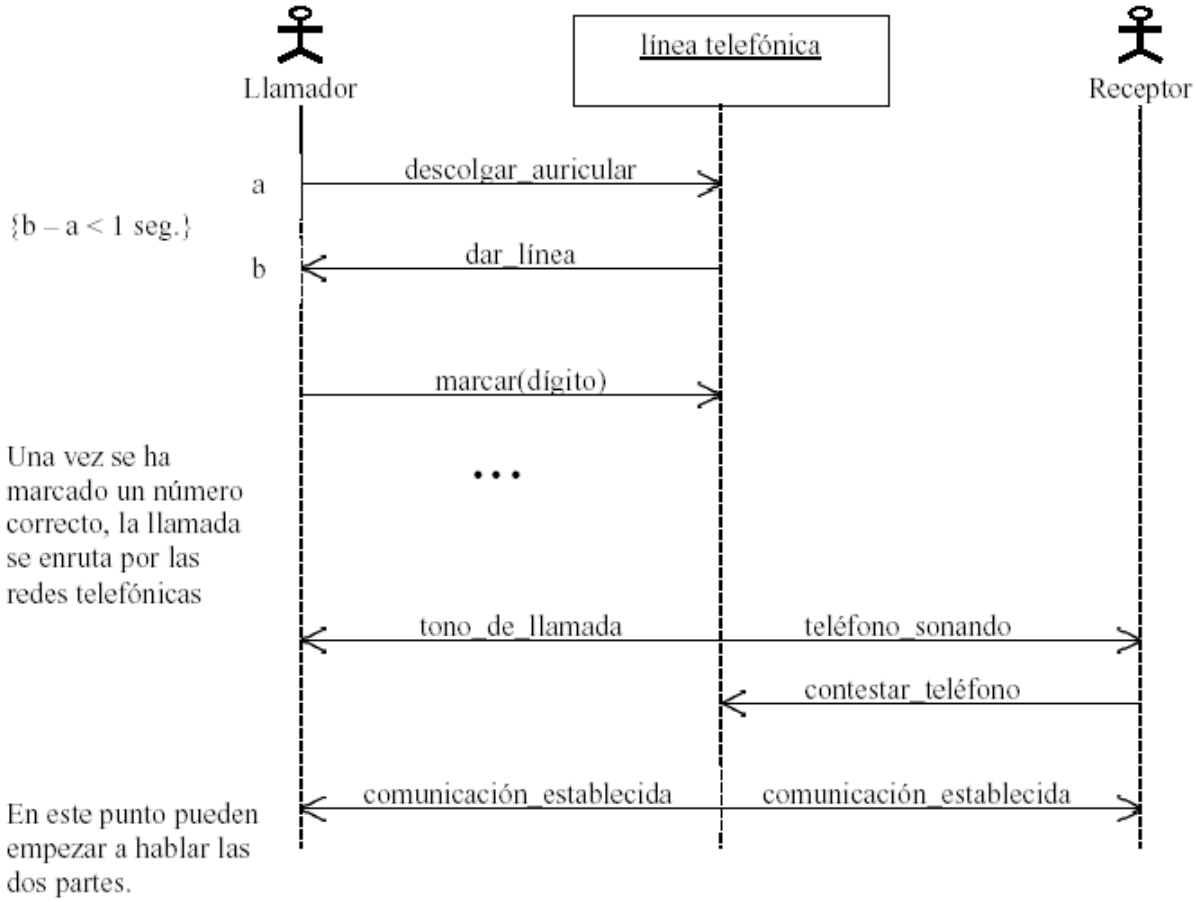
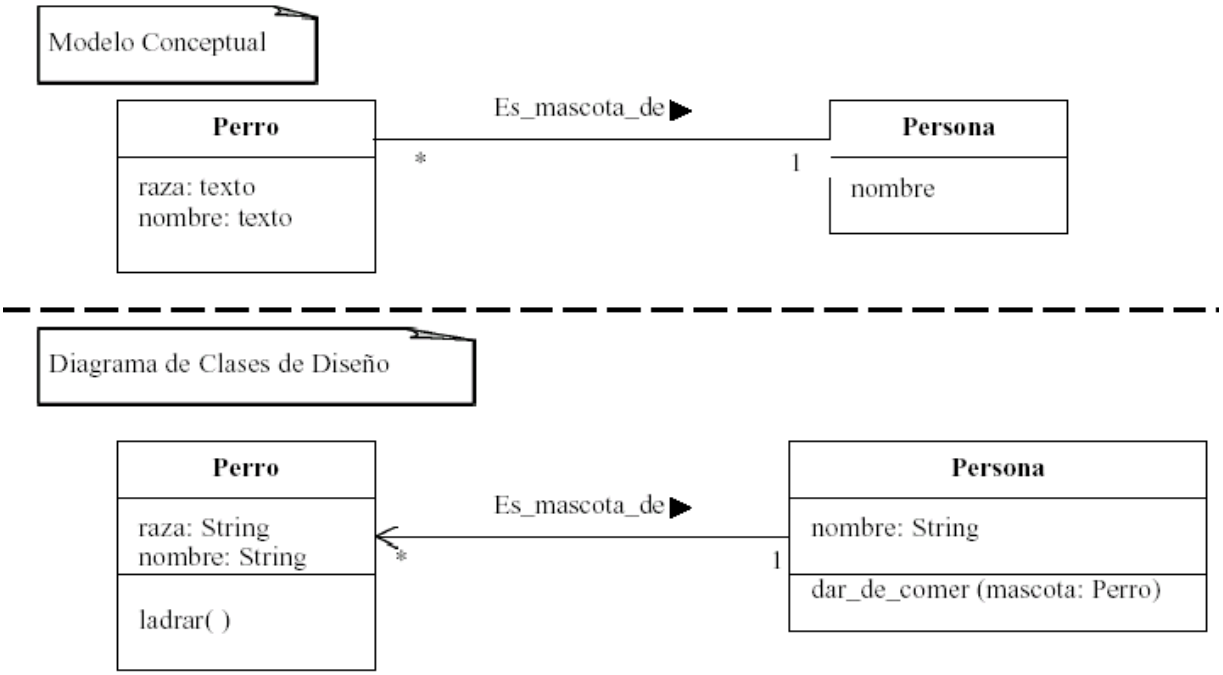


Diagrama de Secuencia

2.2.4 Diagrama de Clases de Diseño

El conjunto de todas las clases usadas, junto con sus relaciones, forma el Diagrama de Clases de Diseño. Muestra la especificación para las clases software de una aplicación. Incluye la siguiente información:

-Clases, asociaciones y atributos.
-Interfaces, con sus operaciones y constantes.
-Métodos.
-Navegabilidad.
-Dependencias.



A diferencia del Modelo Conceptual, un Diagrama de Clases de Diseño muestra definiciones de entidades de software más que conceptos del mundo real.

2.2.4.1 Construcción de un Diagrama de Clases de Diseño

1. Identificar todas las clases participantes en la solución software, analizando los Diagramas de Interacción.
2. Representarlas en un diagrama de clases.
3. Duplicar los atributos que aparezcan en los conceptos asociados del Modelo Conceptual.
4. Añadir los métodos, según aparecen en los Diagramas de Interacción.
5. Añadir información de tipo a los atributos y métodos.
6. Añadir las asociaciones necesarias para soportar la visibilidad de atributos requerida.
7. Añadir flechas de navegabilidad a las asociaciones para indicar la dirección de visibilidad de los atributos.
8. Añadir relaciones de dependencia para indicar visibilidad no correspondiente a atributos.

No todas las clases que aparecían en el Modelo Conceptual tienen por qué aparecer en el Diagrama de Clases de Diseño, tan solo se incluirán aquellas clases que tengan interés en cuanto a que se les ha asignado algún tipo de responsabilidad en el diseño de la interacción del sistema.

En la fase de Diseño se añaden los detalles referentes al lenguaje de programación que se vaya a usar. Por ejemplo, los tipos de los atributos y parámetros se expresarán según la sintaxis del lenguaje de implementación escogido.

2.2.4.2 Navegabilidad

La navegabilidad es una propiedad de un rol (un extremo de una asociación) que indica que es posible "navegar" unidireccionalmente a través de la asociación, desde objetos de la clase origen a objetos de la clase destino. La navegabilidad implica visibilidad, normalmente visibilidad por medio de un atributo en la clase origen. En la implementación se traducirá en la clase origen como un atributo que sea una referencia a la clase destino.

Las asociaciones que aparezcan en el Diagrama de Clases deben cumplir una función, deben ser necesarias, si no es así deben eliminarse.

Las situaciones más comunes en las que parece que se necesita definir una asociación con navegabilidad de A a B son:

-A envía un mensaje a B.
-A crea una instancia B.
-A necesita mantener una conexión con B.

2.2.4.3 Visibilidad

Los atributos y los métodos deben tener una visibilidad asignada, que puede ser:

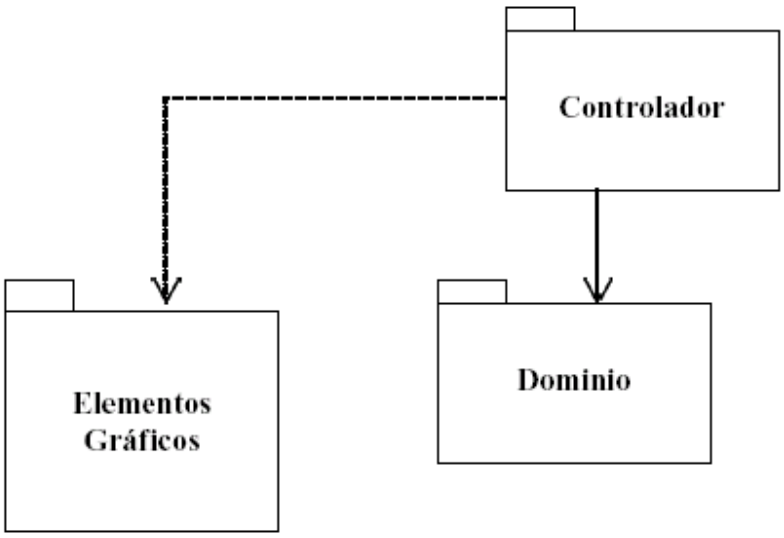
+ Visibilidad pública.
# Visibilidad protegida.
- Visibilidad privada.

También puede ser necesario incluir valores por defecto, y todos los detalles ya cercanos a la implementación que sean necesarios para completar el Diagrama de Clases.

2.2.5 Otros aspectos en el Diseño del Sistema

En el diseño de un sistema es necesario también tomar decisiones a un nivel más alto sobre la descomposición de un sistema en subsistemas y sobre la arquitectura del sistema. Esta parte del Diseño es lo que se denomina Diseño del Sistema. Estas decisiones no se toman de forma distinta en un desarrollo orientado a objetos a como se llevan a cabo en un desarrollo tradicional. Sí hay que tener en cuenta que las posibles divisiones en subsistemas tienen que hacerse en base a las clases definidas en el Diagrama de Clases del Diseño.

Para realizar una división en módulos o paquetes del sistema, se empleará la notación para paquetes que define UML:



2.3 Implementación: Se lleva lo especificado en el diseño a un lenguaje de programación.
2.4 Pruebas: Se llevan a cabo una serie de pruebas para corroborar que el software funciona correctamente y que satisface lo especificado en la etapa de Planificación y Especificación de Requisitos.

3. Instalación: La puesta en marcha del sistema en el entorno previsto de uso.