

# UML

TEORÍA DE CASOS DE USO

DISEÑO DE APLICACIONES | 2021



## Contenido

El modelado en el desarrollo de software	2
¿Qué es un modelo?	2
La importancia de modelar	2
principios de modelado	3
Advertencias en el Modelado	3
lenguaje de modelado unificado (UML)	4
Conceptos básicos sobre UML	4
Breve Reseña Histórica	4
¿Qué es la OMG?	5
Presentación de los diagramas de UML	5
Diagrama de Caso de uso	7
Componentes del Diagrama de Caso de uso	7
Caso de uso	7
Actor	8
Asociación	8
¿Cómo Encontrar Casos de uso?	9
¿Todos los Casos de uso se describen en detalle?	9
Descripción del Caso de Uso	10
Un caso de uso tiene muchas instancias posibles.	11
¿Cómo sabe usted cuándo dejar de escribir un escenario o un caso de uso?	12
Descripción de los casos de uso	12
Lineamientos para el contenido del flujo de eventos:	12
¿Por qué definir actores?	13
¿Cómo encontrar actores?	13
Los actores ayudan a definir los límites del sistema	14
Breve descripción de los actores	14
Generalización de Actores	15
Uso del Diagrama de Caso de uso	15
Casos de uso Concretos y Abstractos	15
Estructurando el Modelo de Caso de uso	16
¿Están los casos de uso relacionados siempre a Actores?	17

## El modelado en el desarrollo de software

El éxito en el desarrollo de software está dado en la medida que se pueda satisfacer a las necesidades de los usuarios con un producto de calidad. El producto principal del trabajo de un equipo de desarrollo no son bonitos documentos, ni reuniones importantes, ni reportes de avance. Más bien, es **un buen software que satisfaga las necesidades cambiantes de sus usuarios** y demás afectados, todo lo demás es secundario.

No obstante, no se debe confundir “secundario” con “irrelevante”. Para producir software de calidad, es necesario involucrar a los usuarios para poder obtener un conjunto de requerimientos. Para desarrollar software de calidad, debemos definir una arquitectura sólida que le de soporte y sea permeable a los cambios. Para desarrollar software rápida, efectiva y eficientemente, con el mínimo de desperdicios y reconstrucción, hay que contar con el equipo de desarrollo adecuado, herramientas y el enfoque apropiado.

### ¿QUÉ ES UN MODELO?

**Un modelo es una simplificación de la realidad.**

Un modelo proporciona los planos del sistema. Estos planos pueden ser detallados o generales. Un buen modelo incluye aquellos elementos que tienen una gran influencia y omite aquellos elementos menores, que no son relevantes para el nivel de abstracción que se desea mostrar.

Todo sistema debe ser descrito desde distintas perspectivas utilizando distintos modelos. En el software, un modelo puede ser de estructura, destacando los elementos que organizan el sistema o de comportamiento, resaltando aspectos funcionales y/o de interacción dinámica.

### LA IMPORTANCIA DE MODELAR

Construimos modelos para comprender mejor el sistema que estamos desarrollando. Por medio del modelado conseguimos cuatro objetivos:

1. Los modelos nos ayudan a visualizar cómo es o queremos que sea un sistema.
2. Los modelos nos permiten especificar la estructura o el comportamiento de un sistema.
3. Los modelos proporcionan plantillas que nos guían en la construcción de un sistema.
4. Los modelos documentan las decisiones que hemos adoptado.

Los modelos son útiles siempre, no sólo para sistemas grandes. No obstante, es cierto que mientras más grande y complejo sea el sistema el modelado se hace más importante, puesto que construimos modelos de sistemas complejos porque no podemos comprender el sistema en su totalidad. Esto es así porque la capacidad humana de comprender la realidad tiene límites y con el modelado se reduce el problema, centrándose en un aspecto a la vez.

### Principios de modelado

El uso de modelos tiene una historia interesante en todas las disciplinas de ingeniería. Esa experiencia sugiere 4 principios básicos de modelado:

1. La elección de qué modelos crear tiene una profunda influencia sobre cómo se aborda un problema y cómo se da forma a su solución.
2. Todo modelo puede ser expresado a diferentes niveles de precisión.
3. Los mejores modelos están ligados a la realidad.
4. Un único modelo no es suficiente. Cualquier sistema no trivial se aborda mejor a través de un pequeño conjunto de modelos casi independientes.

### Advertencias en el modelado

A pesar de las ventajas obvias del modelado existe un factor importante que ejerce influencia en el valor de un modelo: la calidad del modelo en sí mismo. Si la abstracción es incorrecta, entonces obviamente la realidad creada eventualmente fuera de la abstracción, probablemente será incorrecta. Una abstracción incorrecta, no refleja o representa verdaderamente la realidad. Por lo tanto, la calidad del modelo es de inmensa importancia para derivar beneficios de calidad.

El modelado está limitado por las siguientes advertencias:

- **Un modelo es una abstracción de la realidad.** El modelador, dependiendo de sus necesidades, muestra partes que son importantes para una situación particular y deja fuera otras que considera menos importantes. Por lo tanto, *el modelo no es una representación completa de la realidad*. Esto lleva a la posibilidad que el modelo esté sujeto a diferentes interpretaciones.
- **A menos que el modelo sea dinámico, no provee el correcto sentido de la evolución en función del tiempo.** Dado que la realidad es cambiante, es imperativo que el modelo cambie en consecuencia. De otra forma, no podrá transmitir el significado real al usuario.
- **Un modelo puede crearse para una situación específica o para manejar un problema particular.** No hace falta decir que una vez que la situación ha cambiado, el modelo no será relevante.

- **Un modelo es una representación única de posibles múltiples elementos de la realidad.** Por ejemplo: una clase en software es una representación de múltiples objetos, sin embargo, no provee información respecto a volumen o performance.
- **El programador del modelo puede no estar informado de las notaciones y el lenguaje utilizado para expresar el modelo.** El no comprender las notaciones o el proceso puede volver inútil al modelo.
- **Modelar informalmente, sin el debido cuidado y consideración por la naturaleza de los modelos en sí mismos, usualmente provoca confusión en los proyectos y puede reducir productividad.** Por lo tanto, la formalidad en el modelado es necesaria para su éxito.
- **Los modelos deben cambiar con el cambio de la realidad.** Conforme las aplicaciones y sistemas cambian los modelos deben hacer lo mismo para continuar siendo relevantes. Los modelos desactualizados pueden ser engañosos.
- **Los procesos juegan un rol significativo en dirigir las actividades de modelado.** El modelado sin considerar procesos es una práctica potencial riesgosa y debería evitarse.

## LENGUAJE DE MODELADO UNIFICADO (UML)

### Conceptos básicos sobre UML

UML son las siglas para Unified Modeling Language, que en español quiere decir: Lenguaje de Modelado Unificado.

**Es un lenguaje de modelado, de propósito general, usado para la visualización, especificación, construcción y documentación de sistemas Orientados a Objetos.**

Para comprender qué es el UML, basta con analizar cada una de las palabras que lo componen, por separado.

- **Lenguaje:** el UML es, precisamente, un lenguaje, lo que implica que éste cuenta con una sintaxis y una semántica. Por lo tanto, al modelar un concepto en UML, existen reglas sobre cómo deben agruparse los elementos del lenguaje y el significado de esta agrupación.
- **Modelado:** el UML es visual. Mediante su sintaxis se modelan distintos aspectos del mundo real, que permiten una mejor interpretación y entendimiento de éste.
- **Unificado:** unifica varias técnicas de modelado en una única.

## Breve Reseña Histórica

Las raíces del UML provienen de tres métodos distintos: el método de Grady Booch, la Técnica de Modelado de Objetos de James Rumbaugh y “Objetory”, de Ivar Jacobson. Estas tres personalidades son conocidas como “los tres amigos”. En 1994 Booch, Rumbaugh y Jacobson dieron forma a la primera versión del UML y en 1997 fue aceptado por la OMG, fecha en la que fue lanzada la versión v1.1 del UML. Desde entonces, UML atravesó varias revisiones y refinamientos hasta llegar a la versión actual: UML 2.5.

## ¿Qué es la OMG?

La OMG (Object Management Group) es una asociación sin fines de lucro formada por grandes corporaciones, muchas de ellas de la industria del software, como, por ejemplo: IBM, Apple, Oracle y Hewlett-Packard. Esta asociación se encarga de la definición y mantenimiento de estándares para aplicaciones de la industria de la computación.

Podemos encontrar más información sobre la OMG en su sitio oficial: <http://www.omg.org>. UML ha madurado considerablemente desde UML 1.1. Varias revisiones menores (UML 1.3, 1.4 y 1.5) han corregido defectos y errores de la primera versión de UML. A estas le ha seguido la revisión mayor UML 2.0 que fue adoptada por el OMG en 2005.

A partir de la versión 2.0 de UML, se modificó el lenguaje, de manera tal que permitiera capturar mucho más comportamiento (Behavior). De esta forma, se permitió la creación de herramientas que soporten la automatización y generación de código ejecutable, a partir de modelos UML.

Dado que es un modelo en constante revisión y evolución, la última versión formalmente liberada es el UML 2.5.

UML permite aplicar en el desarrollo de software, gran variedad de formas para dar soporte a una metodología de desarrollo de software (tal como el Proceso Unificado Racional o RUP), pero no especifica en sí mismo qué metodología o proceso usar.

UML cuenta con varios tipos de diagramas, los cuales muestran diferentes aspectos de los sistemas que se modelan.

## Presentación de los diagramas de UML

En el siguiente cuadro se muestra una breve descripción de cada uno de los diagramas que integran la versión 2.0 de UML, así como su clasificación.

Clasificación	Diagrama	Descripción
---------------	----------	-------------

<b>Diagrama de Estructura</b>  Estos diagramas modelan la estructura del sistema, por consiguiente, son diagramas estáticos.	<b>Diagrama de Clases</b>	Muestra una colección de elementos de modelado, tales como clases, tipos y sus contenidos y relaciones.	
	<b>Diagrama de Componentes</b>	Representa los componentes que componen una aplicación, sistema o empresa. Los componentes, sus relaciones, interacciones y sus interfaces públicas. Un componente es una pieza de código de cualquier tipo.	
	<b>Diagrama de Objetos</b>	Un diagrama que presenta los objetos y sus relaciones en el momento del tiempo. Un diagrama de objetos se puede considerar como un caso especial, un ejemplo de instanciación de un diagrama de clases.	
	<b>Diagrama de Estructura Compuesta</b>	Representa la estructura interna de un clasificador (tal como una clase, un componente o un caso de uso), incluyendo los puntos de interacción del clasificador con otras partes del sistema.	
	<b>Diagrama de Despliegue</b>	Un diagrama de despliegue muestra cómo y dónde se desplegará el sistema. Las máquinas físicas y los procesadores se representan como nodos. Como los componentes se ubican en los nodos para modelar el despliegue del sistema, la ubicación es guiada por el uso de las especificaciones de despliegue.	
	<b>Diagrama de Paquetes</b>	Un diagrama que presenta cómo se organizan los elementos de modelado en paquetes y las dependencias entre ellos.	
<b>Diagramas de Comportamiento</b>  Diagramas que modelan el comportamiento dinámico del sistema a nivel de clases.  La excepción es el diagrama de casos de uso que modela el comportamiento estático del sistema.	<b>Diagrama de Casos de Uso</b>	Un diagrama que muestra las relaciones entre los actores (que representan el ambiente de interés para el sistema), y los casos de uso.	
	<b>Diagrama de Máquinas de Estado</b>	Un diagrama que ilustra cómo los objetos de una clase, se puede mover entre estados que clasifican su comportamiento, de acuerdo con eventos disparadores de transiciones.	
	<b>Diagrama de Actividades</b>	Representa los procesos de negocios de alto nivel, incluidos el flujo de datos. También puede utilizarse para modelar lógica compleja o paralela dentro de un sistema.	
	<b>Diagramas de Interacción.</b>  Tipo especial de diagramas de comportamiento que modelan la interacción	<b>Diagrama de Comunicación</b>	Es un diagrama que enfoca la interacción entre líneas de vida, donde es central la arquitectura de la estructura interna y cómo ella se corresponde con el pasaje de mensajes. La secuencia de los mensajes



	entre objetos, por consecuencia modelan el comportamiento dinámico del sistema.		se da a través de un esquema numerado de la secuencia.
		<b>Diagrama de Secuencias</b>	Un diagrama que representa una interacción, poniendo el foco en la secuencia de los mensajes que se intercambian, junto con sus correspondientes ocurrencias de eventos en las Líneas de Vida.
		<b>Diagrama de Tiempos</b>	El propósito primario del diagrama de tiempos es mostrar los cambios en el estado o la condición de una línea de vida (representando una Instancia de un Clasificador o un Rol de un clasificador) a lo largo del tiempo lineal. El uso más común es mostrar el cambio de estado de un objeto a lo largo del tiempo, en respuesta a los eventos o estímulos aceptados.
		<b>Diagrama Global de Interacción</b>	Tipo especial de diagramas de comportamiento que modelan la interacción entre objetos, por consecuencia modelan el comportamiento dinámico del sistema.

*Tabla 1 - Diagramas de UML 2.5*

La elección de los modelos a construir varía en cada situación en particular y cada uno de los 13 diagramas de UML 2.5 sirve para modelar aspectos diferentes del software.

### Diagrama de Caso de uso

Un diagrama de Casos de Uso muestra las relaciones entre los actores y los casos de uso. Muestra actores, casos de uso, paquetes de casos de uso y sus relaciones.

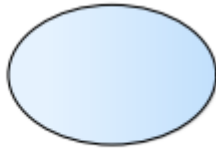
Los diagramas de casos de uso pueden organizarse en paquetes de casos de uso, mostrando sólo lo que es relevante con un paquete particular.

La interacción entre actores no se ve en el diagrama de casos de uso.



## Componentes del Diagrama de Caso de uso

### Caso de uso



#### Nombre del Caso de Uso

Según UML, el nombre del caso de uso debe colocarse debajo de la elipse que lo representa, sin embargo, no todas las herramientas que modelan UML, respetan esta convención, colocando el nombre dentro de la elipse.

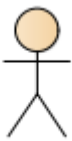
**Un caso de uso registra un contrato entre los involucrados del sistema acerca de su comportamiento, en varias circunstancias, organizándolo por los objetivos de los actores seleccionados.**

El nombre del caso de uso deberá indicar lo que se alcanza por su interacción con el actor. El nombre es una frase verbal, que inicia con un verbo en infinitivo. Puede tener varias palabras y debe ser entendible. No puede haber dos casos de uso con el mismo nombre.

Ejemplos de nombres podrían ser:

- Registrar venta
- Generar comprobante de inscripción.

### Actor



#### Nombre del Actor

Un actor es cualquier entidad que intercambia datos con el sistema, por ejemplo puede ser un usuario u otro sistema (software).

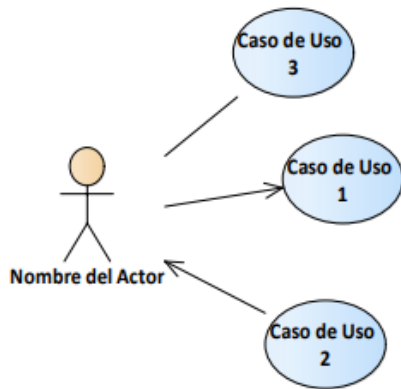
La diferencia entre un actor y una instancia es que el actor representa una clase más que un usuario real. Varios usuarios pueden jugar el mismo rol. En tal caso, cada usuario constituye una instancia de actor.

Dado que un actor representa un rol, el nombre debe ser representativo de ese rol.

Para comprender completamente el propósito del sistema, se debe saber para quién es el sistema.

El mismo usuario puede también actuar como varios actores, es decir, la misma persona puede tomar diferentes roles.

## Asociación



La asociación es una relación que vincula a un actor con un caso de uso.

La asociación es una relación que puede plantearse según las siguientes opciones:

- Actor-Caso de Uso 1: esta navegabilidad se utiliza para indicar que es el actor el que inicia la relación de interacción con el caso de uso.
- Actor-Caso de Uso 2: esta navegabilidad se utiliza para indicar que es el Caso de Uso el que inicia la relación de interacción con el Actor.
- Actor-Caso de uso 3. Se utiliza cuando no está definido aún quien de los dos iniciar la relación de interacción.

La funcionalidad del sistema es definida por diferentes casos de uso, cada uno de los cuales representa un flujo de eventos específico. La descripción de un caso de uso define qué ocurre en el sistema cuando el caso de uso es ejecutado.

La colección de casos de uso constituye todas las maneras posibles de usar el sistema. Se puede obtener una idea de la tarea del caso de uso simplemente observando su nombre.

## ¿Cómo Encontrar Casos de uso?

El siguiente conjunto de preguntas es útil para identificar casos de uso:

- Para cada actor identificado: ¿cuáles son las tareas en las cuales el sistema debería estar involucrado?
- ¿Necesita el actor ser informado acerca de ciertas ocurrencias en el sistema?
- ¿Necesita el actor informar acerca de cambios externos, repentinos?
- ¿El sistema brinda al negocio el comportamiento correcto?
- ¿Pueden ejecutarse todos los aspectos por los casos de uso que se han identificado?
- ¿Qué casos de uso soportará y mantendrán el sistema?
- ¿Qué información debe ser modificada o creada en el sistema?

También son casos de uso, aunque no representan las funciones principales del sistema, los siguientes:

- Inicio y finalización del sistema.
- Mantenimiento del Sistema. Por ejemplo: agregar nuevos usuarios, definir perfiles de usuarios.

- Mantenimiento de los datos almacenados en el sistema, ejemplo: el sistema debe trabajar en paralelo con un sistema anterior, heredado y los datos necesitan sincronizarse entre los dos.
- Funcionalidad necesaria para modificar el comportamiento del sistema.

### ¿Todos los Casos de uso se describen en detalle?

El UML define lineamientos respecto a cómo construir diagramas de casos de uso y una recomendación de complementar los diagramas con descripciones para los casos de uso. No obstante, no define estándares respecto de cómo deben ser esas descripciones.

Frecuentemente habrá en los modelos de casos de uso algunos casos de uso que son tan simples que no necesitan una descripción detallada del flujo de eventos, una descripción general es suficiente. El criterio para tomar esa decisión es que no se vean desacuerdos entre quienes lo leerán o en lo que el caso de uso significa.

### Descripción del Caso de Uso

El propósito de la descripción o narrativa de un caso de uso es contar la historia de cómo el sistema y sus actores trabajan en conjunto para alcanzar una meta específica. Las narrativas de los casos de uso:

- Describen una secuencia de acciones, incluyendo variantes, que un sistema y sus actores pueden ejecutar para alcanzar una meta.
- Se representan como un conjunto de flujos que describen cómo un actor emplea un sistema para alcanzar una meta y lo que el sistema hace para ayudar al actor a alcanzar esa meta.
- Capturan la información de los requisitos necesarios para apoyar las otras actividades del desarrollo.

Las narrativas de los casos de uso se pueden capturar de muchas maneras, incluyendo una Wiki, tarjetas indexadas, documentos de MS Word o dentro de una de las muchas herramientas de administración, de manejo de requisitos o de modelado de requisitos comercialmente disponibles.

Las narrativas de los casos de uso se pueden desarrollar a distintos niveles de detalle, desde un simple resumen, identificando el flujo básico y las variantes más importantes, hasta una especificación completa, altamente detallada, que defina las acciones, entradas y salidas involucradas en la ejecución del caso de uso. Las narrativas de los casos de uso se pueden preparar en los siguientes niveles de detalle:

#### Brevemente Descripta

El nivel más liviano de detalle que sólo captura la meta del caso de uso y el actor que lo inicia. Este nivel de detalle es apropiado para aquellos casos de uso que usted decida no

implementar. Se necesitará más detalle si el caso de uso se va a dividir para su implementación.

#### Con Resumen Básico

El caso de uso se debe resumir para entender su tamaño y complejidad. Este nivel de detalle también posibilita un manejo efectivo del alcance, puesto que el resumen permite que las distintas partes del caso de uso se prioricen unas con otras y, si es necesario, se destinen a distintas versiones. Este es el nivel más liviano de detalle que posibilita que el caso de uso se divida y desarrolle. Es ajustable para aquellos equipos que trabajan en estrecha colaboración con sus usuarios y que son capaces de completar cualquier detalle faltante vía conversaciones y la terminación de los casos de prueba acompañantes.

#### Con Resumen Esencial

Algunas veces es necesario aclarar las responsabilidades del sistema y sus actores mientras se trabaja en el caso de uso. Un resumen básico captura sus responsabilidades, pero no define claramente qué partes del caso de uso son responsabilidad del caso de uso y qué partes son responsabilidad de los actores. En este nivel de detalle, la narrativa se convierte en una descripción del diálogo entre el sistema y sus actores. Es particularmente útil cuando se establece la arquitectura de un nuevo sistema o se intenta establecer una nueva experiencia de usuario.

#### Completamente Descripta

Las narrativas de los casos de uso se pueden usar para proporcionar una especificación bastante detallada de los requisitos, haciéndolas evolucionar hasta su nivel de detalle más completo. El detalle extra puede ser necesario para cubrir la falta de experiencia dentro del equipo, la falta de acceso a los interesados o para comunicar efectivamente requisitos complejos.

Este nivel de detalle es particularmente útil para aquellos casos de uso donde un malentendido del contenido podría tener consecuencias severas de seguridad, finanzas o asuntos legales. También puede ser útil cuando hacemos desarrollo de software externo o por subcontratación.

La narrativa del caso de uso es un producto de trabajo bastante flexible que se puede expandir para capturar la cantidad de detalle necesario para cualquier circunstancia.

Si usted es parte de un equipo pequeño que trabaja de manera colaborativa con el cliente o de un proyecto exploratorio, entonces los resúmenes básicos proporcionarán una forma liviana de descubrir los requisitos. Si usted está trabajando en un entorno más rígido donde hay poco acceso a los expertos, entonces las narrativas con resúmenes esenciales o las completamente descritas se pueden usar para cubrir los vacíos en el conocimiento del equipo. **No todos los casos de uso se deben**

**especificar al mismo nivel de detalle;** no es extraño que los casos de uso más importantes y riesgosos tengan más detalle que los otros. Lo mismo va para las secciones de la narrativa del caso de uso: las partes más importantes, complejas o riesgosas de un caso de uso, frecuentemente, se describen con más detalle que las otras.

**Un caso de uso tiene muchas instancias posibles.**

Un caso de uso puede seguir un número ilimitado, pero numerable de caminos. **Estos caminos representan escenarios que están dentro del caso de uso**, porque responden al mismo objetivo, aunque con algunas variaciones.

El escenario elegido depende de los eventos. Los tipos de eventos incluyen:

- **Entrada desde un actor:** un actor puede decidir, de varias opciones, que hacer después. (Ejemplo: al actor puede decidir ingresar otra botella en una maquina recicladora de botellas o pedir el recibo para finalizar la transacción).
- **Un control de valores o tipos de un objeto o atributo interno:** el flujo de eventos puede diferir si un valor es más o menos grande que un cierto valor. (Ejemplo: en un caso de uso de extracción en un cajero automático, el flujo puede diferir si el actor pide más dinero del que puede extraer de su cuenta).

Los escenarios pueden agruparse, según si cumplen con el objetivo o no, en escenarios o flujos de éxito y escenarios o flujos de fracaso, respectivamente. Dentro de los escenarios de éxito, suele destacarse el que se conoce como “Escenario Feliz”, que es el que describe el flujo de eventos que conforma la manera más sencilla de cumplir con el objetivo.

**¿Cómo sabe usted cuándo dejar de escribir un escenario o un caso de uso?**

Hay dos cláusulas que tienen que ser hechas explícitamente para conseguir los límites apropiados en un caso de uso y un escenario. Para ambos casos las cláusulas son las mismas:

- **Cláusula 1.** Todas las interacciones se relacionan al mismo objetivo.
- **Cláusula 2.** Las interacciones empiezan activando al evento y terminan cuando el objetivo se alcanza o se abandona, y el sistema completa sus responsabilidades con respecto a la interacción.

La Cláusula 2 trae a colación la noción de tener una responsabilidad con respecto a la interacción. Muchos sistemas tienen que anotar cada interacción con un cliente (Ej. los sistemas de banco). Si la cláusula no incluyera este requisito, el guion acabaría sin esa responsabilidad.

### Descripción de los casos de uso

El flujo de eventos de un caso de uso contiene la información más importante derivada del trabajo de modelado de casos de uso. El flujo debería presentar lo que el sistema hace, no cómo está diseñado el sistema para ejecutar el comportamiento requerido.

### Lineamientos para el contenido del flujo de eventos:

- Describir cómo inicia y termina el caso de uso.
- Describir qué datos se intercambian entre el actor y el caso de uso.
- No describir detalles de la interfaz del usuario, las descripciones deben ser con terminología neutra, es decir sin hacer referencia a cómo se va a resolver el comportamiento.
- Describir el flujo de eventos, reforzando la interacción entre actor y caso de uso.
- Describir sólo los eventos que pertenecen a ese caso de uso, y no lo que pasa en otros casos de uso fuera del sistema.
- Evitar terminología vaga tal como “por ejemplo” “etc.” “información”.
- Detalle en el flujo de eventos todos los “que” que deberían responderse, recuerde que los diseñadores de pruebas usarán ese texto para identificar casos de prueba.
- Respete la terminología del dominio del problema, si ha utilizado ciertos términos en otros casos de uso, asegúrese que se usen los mismos términos en el caso de uso que está describiendo y que su significado sea el mismo. Para manejar terminología común, utilice glosarios.

### ¿Por qué definir actores?

Los roles que los actores juegan proveen una perspectiva importante de porqué es necesario el caso de uso y su salida. La razón por la cual un actor particular inicia un caso de uso es casi siempre la razón principal de porqué se necesita el caso de uso en primer lugar. Las diferencias sutiles en el rol que un actor juega pueden producir variaciones en las interacciones del caso de uso. Por ejemplo, determinar que el sistema de procesamiento de préstamos debe responder tanto a clientes individuales como corporativos identifica la necesidad de modelar casos de uso únicos para las necesidades específicas de ambos.

Focalizando en los actores, se puede concentrar en cómo será usado el sistema en lugar de pensar cómo será construido o implementado. También ayuda a refinar y definir los límites del sistema. Finalmente, los actores pueden ser un factor para decidir cómo se llevará a cabo el desarrollo en función de entregas de valor para el negocio.

La definición de actores ayuda a identificar potenciales usuarios que necesitan estar involucrados en el esfuerzo de modelado de caso de uso.

### ¿Cómo encontrar actores?

Revisar las siguientes fuentes de información permitirá encontrar potenciales actores del sistema:

- Diagramas de Contexto u otros modelos existentes que describen el límite del sistema con su entorno. **Buscar entidades externas que tienen interacciones con el sistema.**
- El análisis de involucrados determina que grupo de personas interactuarán con el sistema.
- Especificaciones escritas y otra documentación del proyecto, tal como memos acerca de reuniones con usuarios, ayuda a identificar usuarios que podrán ser actores potenciales.
- Minutas de reuniones de requerimientos. Los participantes de estas sesiones pueden ser importantes, debido a que los roles que ellos representan en la organización pueden interactuar con el sistema.
- Guías de Capacitación y Manuales de Usuarios para procesos y sistemas actuales. Estos manuales y guías están dirigidos a menudo a actores potenciales.

El siguiente grupo de preguntas es útil para tener en mente cuando se identifica actores:

- ¿Quién o qué inicia eventos con el sistema?
- ¿Quién proveerá, usará o quitará información?
- ¿Quién usará esta funcionalidad?
- ¿Quién está interesado en cierto requerimiento?
- ¿En qué parte de la organización será usado el sistema?
- ¿Quién dará soporte y mantendrá el sistema?
- ¿Cuáles son los recursos externos del sistema?
- ¿Qué otros sistemas necesitarán interactuar con este sistema?

Existen algunos aspectos diferentes de los alrededores del sistema que deben representarse como actores separados:

- Usuarios que ejecutan las funciones principales del sistema.
- Usuarios que ejecutan las funciones secundarias del sistema, tal como la administración del sistema.
- Hardware externo que el sistema usa.
- Otros sistemas que interactúan con el sistema.

### Los actores ayudan a definir los límites del sistema

Encontrar actores también significa que se establecen los límites del sistema, lo que ayuda a entender el propósito y alcance del sistema. **Solo aquellos que se comunican directamente con el sistema necesitan ser considerados actores.**



*Si se están incluyendo más roles que los que están alrededor del sistema, está intentando modelar el negocio en el cual el sistema será utilizado, no el sistema en sí mismo.*

Ejemplo: Si está construyendo un sistema para reservas aéreas, para ser usado por un agente de viajes, el actor será el agente de viajes. El pasajero no interactúa directamente con el sistema y por lo tanto no es un actor. Si está construyendo un sistema para reservas aéreas que permitirá a los usuarios conectarse vía Internet, el pasajero interactuará directamente con el sistema y por lo tanto será un actor.

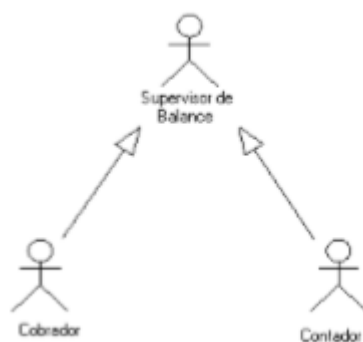
### Breve descripción de los actores

Una breve descripción del actor debería incluir información acerca de:

- A qué o quién representa el actor.
- Porqué es necesario el actor.
- Qué interés tiene el actor en el sistema.
- Características generales de los actores, tales como nivel de experticia (educación), implicaciones sociales (lenguaje), y edad. Estas características pueden influenciar detalles de la interfaz de usuario, tal como fuente y lenguaje.

Estas características son utilizadas principalmente cuando se identifican las clases de interfaz y el prototipo, para asegurar la mejor usabilidad entre la comunidad de usuarios y el diseño de interfaz gráfica.

### Generalización de Actores



Una generalización de un tipo de actor (descendiente) a otro tipo de actor (ancestro) indica que el descendiente hereda el rol que el ancestro puede jugar en un caso de uso.

Varios actores pueden jugar el mismo rol en un caso de uso particular, por ejemplo, los actores cobrador y contador heredan las propiedades del Supervisor de balance, dado que ambos pueden controlar el balance de una

cuenta.

Un usuario puede jugar varios roles en relación con el sistema, lo que significa que un usuario puede corresponder a varios actores. Para hacer más claro el modelo se puede representar al usuario por un actor que hereda a algunos actores. Cada actor heredado representa uno de los roles del usuario relativo al sistema.

### Uso del Diagrama de Caso de uso

No hay reglas estrictas acerca de cómo dibujar un diagrama de caso de uso, pero los siguientes diagramas pueden ser de interés:

- Actores que pertenecen al mismo paquete de caso de uso.
- Un actor y todos los casos de uso con los que interactúa.
- Casos de uso que manejan la misma información.
- Casos de uso usados por el mismo grupo de actores.
- Casos de uso que se ejecutan a menudo con la misma secuencia.
- Casos de uso que pertenecen al mismo paquete de caso de uso.
- Los casos de uso más importantes. Un diagrama de este tipo puede servir como un resumen del modelo.
- Los casos de uso desarrollados juntos, en el mismo incremento.
- Un caso de uso específico y sus relaciones con actores y otros casos de uso.

Es recomendable incluir cada actor, caso de uso y relación en al menos uno de los diagramas. Si hace al modelo de caso de uso más claro, puede formar parte de varios diagramas y/o mostrarlos varias veces en el mismo diagrama.

### Casos de uso Concretos y Abstractos

**Un caso de uso concreto es iniciado por un actor y constituye un flujo de eventos completo.** “Completo” significa que una instancia de un caso de uso ejecuta una operación entera llamada por un actor.

**Un caso de uso abstracto nunca se inicia por un actor. Los casos de uso abstractos son incluidos en, extienden o generalizan otros casos de uso.** Se verán a continuación. Cuando un caso de uso concreto es iniciado, se crea una instancia de caso de uso. Por lo tanto, no se crean instancias separadas desde los casos de uso abstractos.

La distinción entre los dos es importante, debido a que en los casos de uso concretos los actores verán e iniciarán el sistema.

### Estructurando el Modelo de Caso de uso


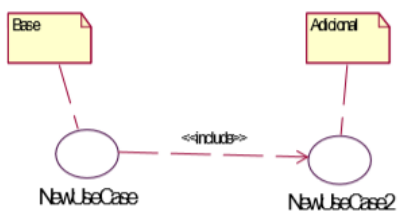
Las razones principales para estructurar el modelo de casos de uso son:

- Para hacer a los casos de uso más fáciles de entender.
- Para particionar comportamiento común descrito en muchos casos de uso.
- Para hacer el modelo de caso de uso más fácil de mantener.

No hay forma de estructurar los casos de uso hasta que no se sabe un poco más de su comportamiento, además de una breve descripción. Al menos se deberá tener una

descripción un poco más amplia para asegurar que las decisiones están basadas en una comprensión suficientemente adecuada del comportamiento.

Para estructurar un caso de uso se tienen tres clases de relaciones:

Tipo de Relación	Definición y características
<p style="text-align: center;"><b>Extensión</b></p> 	<ul style="list-style-type: none"> <li>● Especifica cómo un caso de uso puede insertarse y así extender la funcionalidad de otro.</li> <li>● El caso de uso donde se insertará la extensión debe ser un curso completo en sí mismo.</li> <li>● Se usan para modelar partes optativas, alternativas de un caso de uso, que a su vez tienen un objetivo propio.</li> <li>● Se dibuja con una flecha de línea cortada, cuya dirección va desde el caso de uso de extensión (adicional) al caso de uso base.</li> </ul>
<p style="text-align: center;"><b>Inclusión</b></p> 	<ul style="list-style-type: none"> <li>● Especifica y agrupa comportamiento similar de varios casos de usos, en un caso de uso abstracto, que otros podrán usar.</li> <li>● Se usan cuando su intervención es necesaria para completar un curso completo de eventos.</li> <li>● Se dibuja con una flecha desde el caso de uso concreto o base al caso de uso abstracto (adicional).</li> </ul>

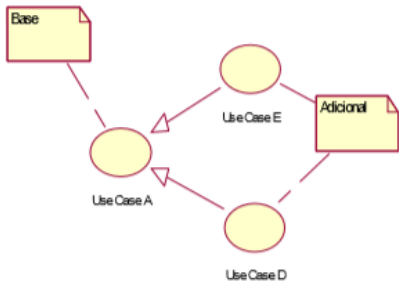
<p style="text-align: center;"><b>Generalización</b></p> 	<ul style="list-style-type: none"> <li>● Se usa cuando los hijos tienen el mismo objetivo y comportamiento que el padre, aunque con variaciones.</li> <li>● Un caso de uso más específico puede especializar a un caso de uso más general.</li> <li>● Una relación de generalización entre casos de usos implica que el caso de uso hijo contiene todos los atributos y secuencias de comportamiento y puntos de extensión definidos para el padre.</li> <li>● Se dibuja con una flecha de línea entera y punta cerrada, desde el caso de uso hijo al padre.</li> <li>● Los casos de uso hijos pueden redefinir el comportamiento heredado del padre.</li> </ul>
--------------------------------------------------------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Tabla 2 - Tipos de relaciones utilizadas para estructurar un diagrama de Casos de Uso

### ¿Cuál es la diferencia entre include y extend?

Cuando relacionamos dos casos de uso con un **include**, decimos que un primer caso de uso "incluye" a otro. Es decir, el segundo caso de uso es parte 'esencial' del primero. Sin el segundo, el primero no podría funcionar bien; o sea no podría cumplir su objetivo.

Por ejemplo en una venta de un artículo, ésta no puede considerarse completa si no se realiza el cobro. El caso de uso "Cobrar Artículo" está incluido en el caso de uso "Vender Artículo", o lo que es lo mismo "Vender Artículo" incluye (<<include>>) "Cobrar Artículo".

En el caso del extend, decimos que un caso de uso "extiende" la funcionalidad de otro, y no es parte esencial del caso de uso base. Es decir, no es absolutamente necesario que se ejecute para que el caso de uso base cumpla con su objetivo. Por ejemplo para el caso de uso "Cobrar Artículo" podría existir un extend a un caso de uso "Acumular puntos" que otorgaría algunos beneficios a los clientes. Pero si dicho caso de uso no se ejecutara, el cobro del artículo podría ejecutarse independientemente.

### ¿Están los casos de uso relacionados siempre a Actores?

La ejecución de cada caso de uso incluye la comunicación con uno o más actores. Una instancia de caso de uso siempre es iniciada con un actor que le pide al sistema que haga algo. Esto implica que cada caso de uso debería tener asociaciones de comunicación con actores. La razón de esta regla es forzar al sistema a proveer solo la funcionalidad que los usuarios necesitan y nada más. Tener casos de uso que nadie requiere es un indicador de que hay algo erróneo en el modelo de caso de uso o en los requerimientos.

Sin embargo, existen algunas excepciones a esta regla:

- Si un caso de uso es abstracto (no instanciable separadamente), su comportamiento puede no incluir interacción con un actor.
- Un caso de uso hijo en una relación de generalización, puede no tener actor asociado, si el caso de uso padre describe toda la comunicación con el actor.
- Un caso de uso base en una relación de inclusión puede no tener un actor asociado, si el caso de uso de inclusión describe toda la comunicación con el actor.
- Un caso de uso que es iniciado de acuerdo con una planificación, lo que significa que el reloj es el iniciador. El reloj del sistema es un evento interno y el caso de uso no es iniciado por un actor. Sin embargo, por claridad, se puede usar un actor ficticio “Tiempo” para mostrar cómo es iniciado el caso de uso en los diagramas de caso de uso.