# Casos de utilización y escenarios

**Esta técnica permite disminuir uno de los riesgos más comunes en un proyecto: la mala delimitación de alcances, debido a requisitos y espectativas mal especificados**

*Este artículo apareció publicado originalmente en dos partes en **Computerworld México**, Mayo 25 y Junio 2 de 1998. En la presente versión (junio de 1999) he hecho algunas adecuaciones para que pueda ser aprovechado no sólo para proyectos informáticos, sino para un ámbito mucho más amplio.*

Resumen de las características de los casos de utilización:

- Definen las principales formas en que el producto de un proyecto va a ser utilizado
- Permiten delimitar requisitos y espectativas
- Ayudan a detectar el comportamiento típico y las situaciones especiales
- Adicionalmente, sirven de base para los guiones de pruebas
- Existe una amplia convergencia sobre su utilidad en las áreas de informática y rediseño de procesos

Por Jaime González

Los riesgos que se deben considerar primero en un proyecto son los relativos al alcance; es decir, qué es lo que realmente nos están solicitando, y cómo lo podemos delimitar para evitar que crezcan sin control tanto los recursos como el tiempo necesario para la terminación. Podemos apreciar la magnitud del problema con el caso del proyecto otorgado en 1988 por la Agencia Federal de Aviación de los EUA para la renovación de sus sistemas de control de vuelos. Durante el desarrollo de éste, la documentación de las adiciones y cambios se acumuló en un altero equivalente a seis metros de alto, y para 1994 el costo adicional estimado alcanzaba ya los miles de millones de dólares.

Sabemos por experiencia que en los primeros encuentros con quienes nos están solicitando la realización del proyecto sólo obtendremos los lineamientos generales, y que conforme avance el tiempo los requerimientos se irán refinando, y que van a sufrir un incesante proceso de modificación. Más adelante aparecerán otros problemas, tales como decidir cual va a ser la arquitectura tecnológica más adecuada para resolver el proyecto, pero por el momento tenemos suficiente en nuestras manos con la especificación y el control de los requerimientos.

Por lo tanto, podemos definir de la siguiente manera nuestro problema inmediato:

1. Cómo podemos documentar, de una manera sencilla, qué es lo que tienen en mente quienes nos están solicitando el proyecto, así como cuáles son las espectativas de los usuarios y de los beneficiarios.

2. Cómo podemos ayudar a todos ellos a definir y a concretar sus requerimientos.

3. Cómo representar el alcance y el comportamiento que deberá contemplar la obra (o el sistema) que vamos a desarrollar en base a estos requerimientos. Esta documentación, una vez acordada con directivos y usuarios, es indispensable para controlar los cambios con respecto a los objetivos originales, en forma tal que las modificaciones a los calendarios de entrega o a los presupuestos se pueden hacer de común acuerdo.

Con frecuencia escuchamos el argumento de que lo anterior no aplica a proyectos con un alto contenido de creatividad, ya que nadie puede delimitar los alcances de un nuevo invento. Pero a pesar

de que el contexto es diferente, en esencia el problema es el mismo: cómo podemos representar qué es lo que se espera que haga el producto (edificio, obra pública, sistema…) resultado de nuestro proyecto.

Por lo menos en el área informática, la forma más común para recoger actualmente los requerimientos está representada por la creación de prototipos y la elaboración directa del producto, sin un modelado previo del sistema y sin seguir un método. Con el objeto de producir resultados rápidamente, generalmente se crean las cuatro o cinco ventanas o pantallas más importantes y se les dota de una mínima funcionalidad. Esta versión reducida del sistema se utiliza como vehículo de comunicación y retroalimentación con el usuario, y también juega el papel de depósito de los requerimientos.

Esta forma de trabajo puede permitir el desarrollo de un sistema cuando la solución está muy cercana al problema planteado; sin embargo, conforme la complejidad va en aumento esta programación salvaje se estrella contra el hecho de que la improvisación no basta para solucionar la enorme cantidad de interrelaciones necesarias entre los componentes del sistema. El prototipeo es una herramienta muy poderosa de comunicación con el usuario, pero no puede sustituir a la especificación de requerimientos, especialmente en proyectos donde el tiempo o los recursos se encuentran limitados. Tampoco puede sustituir a la diagramación, que hace posible la comunicación entre los diversos participantes (entre los cuales debemos contar a quienes se encargarán del mantenimiento del sistema en el futuro).

Recuerdo una experiencia a principios de los años noventa, cuando un usuario nos solicitó que a un inventario de recursos humanos le incluyéramos, por favor, las tallas para los uniformes de los empleados. El desarrollador, deseoso de satisfacer al cliente, incluyó atributos para estas tallas. Más tarde, cuando estábamos demostrando las primeras versiones del sistema, se nos solicitó que se generara un reporte para que el usuario no tuviera que estar copiando las medidas, una por una, desde la pantalla del sistema hacia su forma de solicitud de uniformes. El desarrollador programó entonces un reporte simple. Más adelante nos explicaron que este reporte estaba mal porque la compañía manejaba varios tipos de uniformes, que variaban de acuerdo a si se trataba de empleados o de personal de producción, de acuerdo también a la época del año, y que, por contrato, había quienes tenían derecho a distintos aditamentos como gorras, guantes y botas... Al llegar aquí el problema se tornó en una pesadilla que impedía la liberación de un sistema con el que, por lo demás, el directivo que lo había encargado y los demás usuarios estaban satisfechos.

Este fue un típico caso de un cambio rampante en el alcance del proyecto. Francamente, es poco profesional culpar al usuario o simplemente suponer que actuó de mala fe. Mejor, debemos reflexionar seriamente en cómo resolver nuestros riesgos relativos a requerimientos. El mundo de la ingeniería de sistemas ha venido convirgiendo cada vez más ampliamente en una solución: la técnica de *use cases* (casos de utilización, o técnica de escenarios), que es hoy la forma por excelencia para representar el comportamiento y el alcance de una amplia gama de situaciones (sistemas informáticos, áreas de negocios, instituciones públicas…).

**Un ejemplo simple**

Antes que dar una definición, podemos poner un ejemplo muy sencillo que hemos tomado prestado de Ivar Jacobson, creador original de esta técnica. (Valga aclarar que el desarrollo del ejemplo y su diagramación son responsabilidad exclusiva del autor del presente artículo.)

El primero caso es el de una llamada telefónica: el usuario levanta el auricular, espera el tono de marcar, marca el número con el que se desea comunicar, obtiene el tono de “llamando”, su llamada es contestada, realiza su conversación, y finalmente cuelga el auricular. Con este ejemplo en mente podemos hacer una primera definición: un caso de utilización es una secuencia completa de eventos, desde el principio hasta el final de un proceso, vista desde la perspectiva del usuario. Podemos decir también que se trata de representar una típica interacción del usuario con el sistema, sin preocuparnos de la estructura interna del mismo.

A los usuarios que interactúan con casos de utilización se les denomina *actores*. Y podemos con toda confianza decir que se trata de actores y actrices que están representando un determinado papel en

uno de los escenarios de uso. De hecho, el primer paso para identificar un caso consiste en identificar a los actores. El servicio (o el beneficio) que esperan obtener estos actores es lo que define a nuestros *use cases.*

Los actores generalmente son personajes humanos, pero también pudiera tratarse de otros sistemas o dispositivos: en nuestro ejemplo, el usuario pudiera ser un FAX.

Es importante hacer notar que la llamada local exitosa representa una *secuencia básica* de eventos: es el primer caso que vamos a representar. Esta es la secuencia más importante a resolver y es la que va a guiar todo el desarrollo posterior, desde el análisis hasta las pruebas de aceptación. Por poner otro ejemplo, en la utilización de un cajero automático esta secuencia básica estaría representada por la solicitud y el retiro de efectivo.

Es indispensable detectar estas secuencias básicas, y considerar que algunos sistemas llegan a tener más de una. En el uso del teléfono tenemos los casos de la recepción de la llamada por parte del interlocutor, y también la elaboración de los recibos de cobro por el uso del teléfono. Y vamos a tener que definir actores diferentes para cada uno de estos casos.

Veamos cómo representamos estos tres casos:

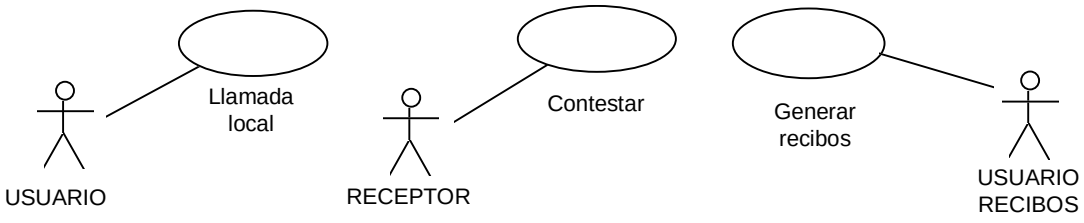


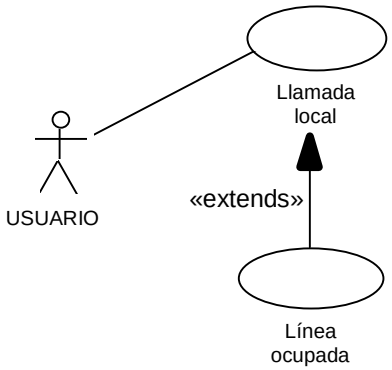

**Diagrama con los casos básicos de un servicio telefónico**

Tenemos en escena tres actores: el usuario del servicio, el receptor de la llamada y el usuario que elabora los recibos. Es importante hacer notar que los actores y actrices representan un determinado papel, y no necesariamente a una persona física determinada. En nuestro ejemplo, una misma persona podría estar jugando el papel del usuario que se encuentra generando los recibos, y podría también ser el receptor de una llamada telefónica, o el usuario que realiza la llamada. Por esto, no sería correcto identificar a un actor con un nombre propio.

Los casos de utilización quedan representados en forma de elipses. Las líneas entre los actores y los casos de utilización representan una relación de interacción con el sistema. La sencillez de este tipo de diagramas nos permite elaborarlos con lápiz y papel, aunque muchas herramientas CASE (*Computer Aided Software Engineering*) cuentan con la capacidad de representar este tipo de diagrama.

**Extensiones**

Las secuencias básicas tienen variantes y situaciones de error, como sucedería en nuestro ejemplo cuando el actor que juega el papel de usuario recibe el tono de “ocupado”. A estas variantes las podemos considerar extensiones de los casos de utilización, y las representamos de la siguiente manera:




**«extends» relaciona el caso con una de sus variantes**

Hemos introducido ahora un nuevo tipo de relación, denominada «extends». “Línea ocupada” abarca los eventos que suceden después que el usuario escucha un tono de “ocupado” después de marcar el número con el que se desea comunicar. La relación «extends», entonces, se interpreta como que uno o varios eventos deben incluirse en un *punto de inserción* preciso de la secuencia correspondiente a “Llamada local”. Todas las relaciónes «extends» tienen implícita una condición para su ejecución. En este caso, la condición consiste en que la línea se encuentra ocupada al momento de llamar.

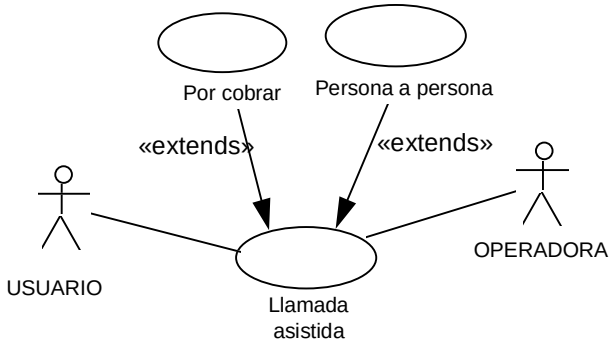
Al elaborar nuestro modelo inicial no debe preocuparnos el especificar detalles tales como el punto de inserción, ya que simplemente estamos tratando de recoger requerimientos y representarlos de manera simple. Lo más probable es que nuestro modelo vaya a sufrir muchos cambios, y por lo tanto conviene esperar a que se estabilice un poco. Aún personas muy experimentadas en el modelado tienen que modificar sus diagramas antes de llegar a producir una versión que sea aceptable tanto para el usuario como para el equipo de desarrollo. Por lo mismo, durante la elaboración del modelo inicial es importante concentrarnos en la secuencia completa y no en detalles como las formas de entrada y salida. Nuestro modelo debe ser visto como una especie de armazón viviente, que se va a ir recubriendo con más y más substancia conforme vamos avanzando en el desarrollo.

Cada caso de utilización tiene instancias concretas de ejecución. Podemos diferenciar una instancia de otra porque pueden ocurrir a diferentes horas del día, o porque las realizó una persona diferente, o porque la llamada se hizo a un número diferente... Estas instancias no quedan descritas ni en los diagramas ni en sus secuencias de eventos (ya que la labor de describirlas sería copiosa, para decir lo menos), y son lo que podríamos denominar *escenarios* de utilización del sistema. En términos rigurosos, un caso de utilización representa una generalización (concentración de elementos comunes) de muchos escenarios particulares. Algunos de estos escenarios, sin embargo, sí nos pueden servir posteriormente para describir los guiones de las diferentes pruebas a las que vamos a someter al sistema.

### Control del alcance y simplificación

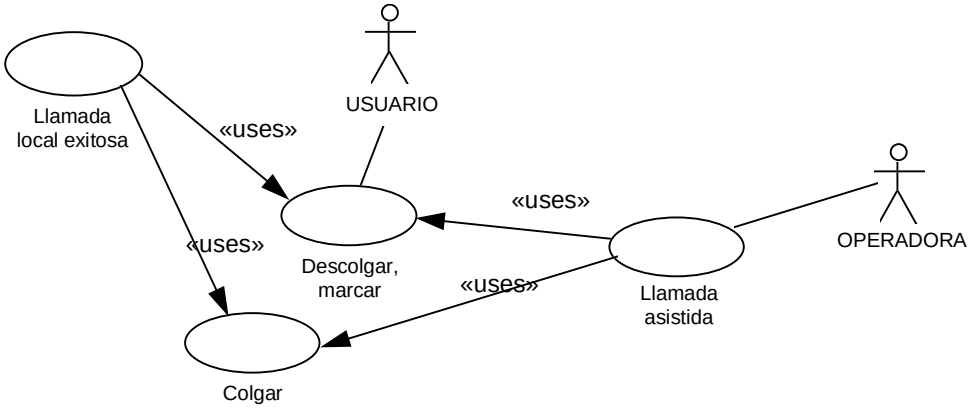
Además de las secuencias básicas puede haber otros casos de utilización que pueden ser parte de la funcionalidad que se desea alcanzar. Este sería el caso de una llamada asistida por operadora, o de una llamada para pedir la hora exacta. Al llegar a este punto es donde debemos considerar qué tanta funcionalidad deseamos incluir en la versión del sistema que estamos construyendo. Por ejemplo, para asegurar cumplir con los tiempos de entrega esperados para el proyecto podríamos acordar incluir en la primera versión sólo las llamadas asistidas por operadora y posponer la realización del servicio de hora exacta para una versión futura. Es en este tipo de decisiones donde podemos aprovechar el modelo para llegar a acuerdos con usuarios y directivos.

Para plasmar el alcance acordado dibujamos los elipses correspondientes a los casos de utilización que hemos decidido incluir en nuestro modelo, con sus extensiones significativas. La llamada asistida por operadora quedaría de la siguiente manera:




**Representación de nuevos casos**

Al representar estos nuevos casos podemos notar que tienen ciertas secuencias en común con los *use cases* que ya tenemos representados. En el ejemplo, “Llamada local exitosa” tiene secuencias de eventos en común con “Llamada asistida”: el usuario del aparato telefónico levanta el auricular, espera el tono de marcar, marca el número... Para evitar tener que repetir estos eventos en cada caso, creamos dos nuevas secuencias, a las que denominaremos “Descolgar, marcar” y “Colgar”, y las representamos como secuencias comunes de la siguiente manera:

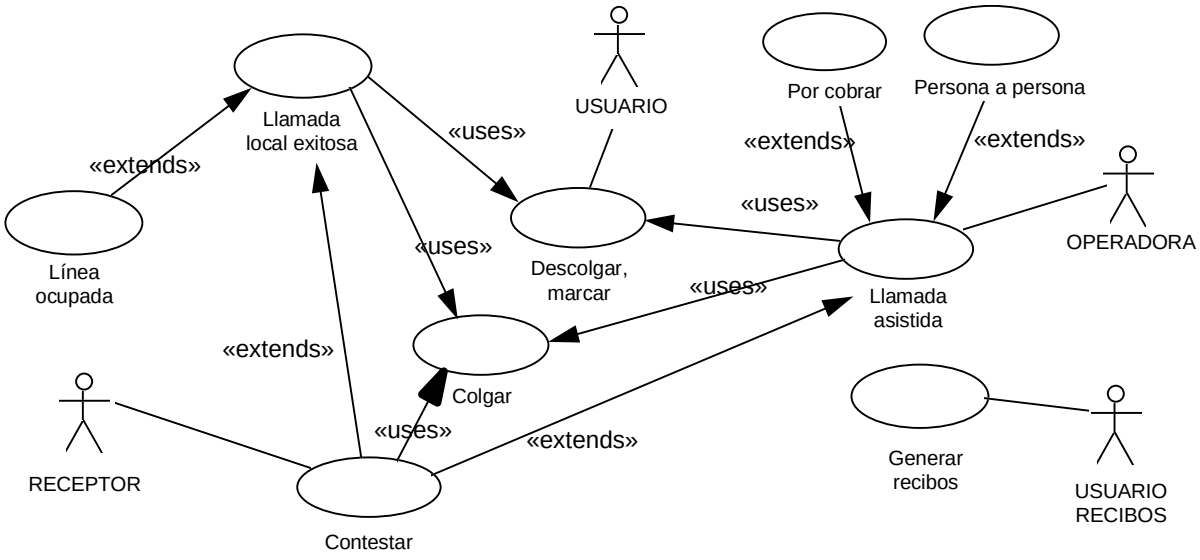



**Simplificación de secuencias comunes mediante «uses»**

Como se puede observar, hemos introducido ahora la relación «uses», que significa que el caso de utilización que está en el origen de la flecha va a incorporar en un punto de inserción preciso la secuencia indicada en el destino. En el ejemplo, la llamada local exitosa y la llamada asistida van a incorporar la secuencia “Descolgar, marcar”, van realizar su propia secuencia de eventos, y finalmente van a terminar con “Colgar”. El orden de ejecución de estas secuencias y los puntos de inserción deberán ser especificados una vez que el modelo se estabilice. Los casos de utilización que “usan” a otros casos puede tener sus propias secuencias, incluso intercaladas con las secuencias “usadas”; sin embargo, no es válido emplear la relación «uses» para representar el uso parcial de otras secuencias.

En nuestro ejemplo, tanto la “Llamada local exitosa” como la “Llamada asistida” deben usar *toda* la secuencia “Descolgar, marcar” y *toda* la secuencia de “Colgar”.

Con esto estamos listos para armar en un solo diagrama nuestros casos de utilización:




**El conjunto queda armado**

Podemos apreciar que la secuencia “Colgar” ahora es compartida por “Llamada local exitosa”, por “Contestar” y por “Llamada asistida”. Y podemos apreciar también que “Contestar” es una extensión común a “Llamada local exitosa” y a “Llamada asistida”.

Se nos pudiera antojar descomponer algunas secuencias en elementos más pequeños aún, como podría ser el caso de descomponer “Descolgar, marcar”, pero no es conveniente crear demasiados elementos en el diagrama: conforme se crea un mayor número de relaciones la complejidad va aumentando en forma exponencial.

Es perfectamente válido (y muchas veces necesario) crear varios diagramas para lograr modelar el alcance de una aplicación. En el caso de sistemas grandes es posible modelar las dependencias entre “paquetes” de *use cases* (para lo cual el estándar Unified Modeling Language cuenta con diagramas especiales, que esperamos poder describir en otra ocasión). Por el momento, podemos sugerir una serie limitada de diagramas particulares que en conjunto representen la funcionalidad y el alcance de la aplicación a desarrollar.

La descripción de la secuencia de eventos de cada *use case* puede hacerse mediante un texto muy breve, o bien mediante diagramas de interacción (que también esperamos describir en otro artículo).

### ¿Qué tantos casos debemos modelar?

Del ejemplo proporcionado podemos inferir que los casos de utilización tienen un ámbito muy amplio de aplicación. De hecho, Jacobson recomienda esta técnica no sólo para aplicaciones de tecnología informática, sino en general para modelar procesos en empresas e instituciones. Sin embargo, es necesario considerar que las situaciones de error y las variantes en un sistema representan un espacio de posibilidades inmenso, que simplemente no es posible modelar. La forma de evitar empantanarnos en una “parálisis por análisis” consiste, entonces, en encontrar y modelar los casos que son significativos para definir y acotar el sistema.

Los casos de utilización pueden variar de tamaño, desde muy pequeños hasta muy extensos. Para lograr mayor claridad en su exposición y para facilitar el mantenimiento del modelo es preferible no llegar a una granularidad demasiado fina. Es necesario tener en mente que los diagramas (y en general todos los modelos) son sólo un medio para ayudar al verdadero fin que perseguimos, que es la satisfacción de los usuarios y beneficiarios. El criterio de aceptación por parte de éstos nunca estará normado por la brillantez de nuestros dibujos, sino por la eficacia y la calidad de los productos de nuestros proyectos.

**Notas bibliográficas y referencias en la red**

Sobre el caso de la Agencia Federal de Aviación (FAA) de los EUA, ver el artículo “Aging Airways” de Gary Stix, en *Scientific American*, May 1994, p. 70.

En su artículo "A growing consensus on use cases", publicado en la edición de marzo-abril de 1995 del *Journal of Object Oriented Programming* (*JOOP*), Magnus Christerson e Ivar Jacobson documentaron la convergencia de los notables metodólogos Grady Booch, Kenneth Rubin, Rebeca Wirfs-Brock y James Rumbaugh en torno a la utilidad de los *use cases.* Ver **http://www.sigs.com/publications/docs/joop**.

El libro de Martin Fowler y Kendall Scott, *UML Distilled* (Addisson-Wesley, Reading, Mass., 1997) contiene un capítulo breve y útil sobre *use cases*.

El texto clásico sobre el tema es el de Jacobson, Christerson, Jonsson y Overgaard*: Object Oriented Software Engineering: A Use Case Driven Approach* (Addison-Wesley and ACM Press, Reading, Mass., 1992). Ivar Jacobson, Maria Ericsson y Agneta Jacobson ofrecen una profundización del tema en las páginas 97 a 149 de *The Object Advantage. Business Process Reengineering with Object Technology*. Addison-Wesley and ACM Press, 1994.

La idea consistente en que no es posible modelar todas las situaciones de error y variantes de un sistema está tomada del libro de Ian Graham, *Métodos orientados a objetos* (Segunda Edición. Addisson-Wesley / Díaz de Santos. Wilmington, Delaware, 1996), página 325.

Los documentos sobre Unified Modeling Language adoptados por el Object Management Group pueden obtenerse de:
**http://www.omg.org/library/schedule/Technology_Adoptions.htm#tbl_UML_Specification**

Para obtener una descripción formal de los elementos que componen los *use cases* y su semántica, ver el documento **UML Semantics**, páginas 91 a 98 en la versión 1.1.