1. Análisis de requerimientos para casos de uso

Para realizar un caso de uso es primordial tener en cuenta los requerimientos. A continuación se explica su definición y características.

¿Qué son requerimientos?

Según el glosario de la IEEE, que corresponde a las siglas de (Institute of Electrical and Electronics Engineers) en español, Instituto de Ingenieros Eléctricos y Electrónicos, un requerimiento es:

- Una condición o necesidad de un usuario para resolver un problema o alcanzar un objetivo.
- Una condición o capacidad que debe estar presente en un sistema o componentes de sistema para satisfacer un contrato, estándar, especificación u otro documento formal.

Analizando las definiciones anteriores, un requerimiento es una descripción de una condición o capacidad que debe cumplir un sistema, ya sea derivada de una necesidad de usuario identificada, o bien, estipulada en un contrato, estándar, especificación u otro documento formalmente impuesto al inicio del proceso.

Los requerimientos deben ser:

- Especificado por escrito: como todo contrato o acuerdo entre dos partes.
- Posible de probar o verificar: si un requerimiento no se puede comprobar, entonces ¿cómo se sabe si se cumplió con él o no?
- Conciso: un requerimiento es conciso si es fácil de leer y entender. Su redacción debe ser simple y clara para aquellos que vayan a consultarlo en un futuro.
- Completo: un requerimiento está completo si no necesita ampliar detalles en su redacción, es decir, si se proporciona la información suficiente para su comprensión.

- Consistente: es consistente si no es contradictorio con otro requerimiento.
- No ambiguo: un requerimiento no es ambiguo cuando tiene una sola interpretación. El lenguaje usado en su definición, no debe causar confusiones al lector.

Los requerimientos pueden dividirse en:

Los requerimientos de software pueden dividirse en dos categorías: requerimientos funcionales y no funcionales.

Los requerimientos funcionales son los que definen las acciones que el sistema será capaz de realizar, describen las transformaciones que el sistema hace sobre las entradas para producir salidas. Es importante que se describa el ¿qué? y no el ¿cómo? se deben hacer esas transformaciones. Estos requerimientos al tiempo que avanza el proyecto de software se convierten en los algoritmos, la lógica y gran parte del código del sistema.

Por otra parte los requerimientos no funcionales tienen que ver con características que de una u otra forma puedan limitar el sistema, como por ejemplo, el rendimiento (en tiempo y espacio), interfaces de usuario, fiabilidad (robustez del sistema, disponibilidad de equipo), mantenimiento, seguridad, portabilidad, estándares, entre otras.

Dificultades para definir los requerimientos

Durante la etapa de especificación de requerimientos se pueden presentar muchos inconvenientes los cuales son importantes de identificar y prevenir, a continuación se presenta un listado con los problemas más comunes en este proceso:

- Los requerimientos no son obvios y vienen de muchas fuentes.
- Son difíciles de expresar en palabras (el lenguaje es ambiguo).
- La cantidad de requerimientos en un proyecto puede ser difícil de manejar.
- Un requerimiento puede cambiar a lo largo del ciclo de desarrollo.
- El usuario no puede explicar lo que hace.
- Tiende a recordar lo excepcional y olvidar lo rutinario.
- Se habla de lo que no funciona.
- Los usuarios tienen distinto vocabulario a los desarrolladores.
- Usan el mismo término con distinto significado.

Recomendaciones para la toma de requerimientos

Para realizar la toma de requerimientos se deben tener en cuenta unas recomendaciones, pues la tarea de análisis de estos, es un proceso de descubrimiento y refinamiento. Se analizan y asignan a los distintos elementos de los programas las soluciones alternativas.

El ingeniero desarrollador y el cliente tienen un papel muy activo en la especificación de requerimientos. El cliente intenta explicar su idea, algo confusa de la función y comportamiento de los programas en detalles concretos, el ingeniero desarrollador del software actúa como interrogador, consultor y el que resuelve los problemas. La tarea de análisis parece una labor sencilla pero se debe tener mucho cuidado, pues lo que el cliente trata de expresar no siempre es lo mismo que el ingeniero entiende, por ello se debe tener una estrecha comunicación para comprender todos los requerimientos que se están tomando, esto evita los cambios por mala interpretación o falta de información.

Los requerimientos de un sistema de software, cuando se ven en su conjunto son extensos y detallados, y además contienen múltiples relaciones entre sí.

El análisis de requerimientos facilita al ingeniero de sistemas especificar la función y comportamiento de los programas, indicar la interfaz con otros elementos y establecer las ligaduras de diseño que debe cumplir el programa, también permite refinar la asignación de software y representar el dominio de la información que será tratada por el sistema y permite diseñar la representación de la información y las funciones que pueden ser traducidas en datos, arquitectura y diseño procedimental. Finalmente, la especificación de requerimientos suministra al técnico y al cliente, los medios para valorar la calidad de los programas.

Tareas del análisis

El análisis de requerimientos puede dividirse en cuatro áreas:

1. Reconocimiento del problema.
2. Evaluación y síntesis.
3. Especificación.
4. Revisión. (Arias, 2006, pp. 2-3)

Técnicas de recopilación de información

Los analistas utilizan diferentes técnicas con el fin de recolectar los datos para llevar a cabo un proyecto y ver su viabilidad. Los métodos más utilizados son las entrevistas, cuestionarios, inspección de registros, observación entre otros. Cada uno tiene sus ventajas y desventajas. Generalmente, se utilizan dos o tres para asegurar una investigación completa.

1. **Entrevistas y cuestionarios:** las entrevistas y cuestionarios se emplean para reunir información proveniente de personas o de grupos. Durante la entrevista, el analista conversa con el encuestado; el cuestionario consiste en una serie de preguntas relacionadas con varios aspectos de un sistema.

Por lo general, los encuestados son usuarios de los sistemas existentes o usuarios en potencia del sistema propuesto. En algunos casos, son gerentes o empleados que proporcionan datos para el proyecto propuesto o que serán afectados por él. El éxito de esta técnica, depende de la habilidad del entrevistador y de su preparación para la misma.

2. **Sistemas existentes:** esta técnica consiste en analizar distintos sistemas ya desarrollados que estén relacionados con el proyecto a ser construido. Por un lado, se pueden analizar las interfases de usuario, observando el tipo de información que se maneja y cómo es utilizada, por otro lado también es útil estudiar las distintas salidas que los sistemas producen (listados, consultas, etc.), porque siempre pueden surgir nuevas ideas sobre la base de estas.

3. **Lluvia de ideas:** este es un modelo que se usa para generar ideas. La intención en su aplicación es la de generar la máxima cantidad posible de requerimientos para el sistema. No hay que detenerse en pensar si la idea es o no del todo utilizable. La intención de este ejercicio es generar en una primera instancia muchas ideas. Luego, se irán eliminando en base a distintos criterios como, por ejemplo, "caro", "impracticable", "imposible", etc.

4. **Prototipos:** durante la actividad de extracción de requerimientos, puede ocurrir que algunos de ellos no estén demasiado claros o que no se esté muy seguro de haber entendido correctamente los obtenidos hasta el momento, todo esto puede llevar a un desarrollo poco eficaz del sistema final. Entonces, para validar los requerimientos hallados, se construyen prototipos.

Los prototipos son simulaciones del posible producto, que luego son utilizados por el usuario final, permitiendo conseguir una importante retroalimentación en cuanto a si el sistema diseñado con base a los requerimientos recolectados le permite al usuario realizar su trabajo de manera eficiente y efectiva. El desarrollo del prototipo comienza con la captura de requerimientos.

Desarrolladores y clientes se reúnen y definen los objetivos globales del software, identifican todos los requerimientos que son conocidos y señalan áreas en las que será necesaria la profundización en las definiciones. Luego de esto, tiene lugar un “diseño rápido”, que se centra en una representación de aquellos aspectos del software que serán visibles al usuario (por ejemplo, entradas y formatos de las salidas). El diseño rápido lleva a la construcción de un prototipo.

5. **Casos de uso:** los casos de uso son una técnica para especificar el comportamiento de un sistema. Permiten entonces describir la posible secuencia de interacciones entre el proyecto y uno o más actores, en respuesta a un estímulo inicial proveniente de un actor, es una descripción de un conjunto de escenarios, cada uno de ellos comenzado con un evento inicial desde un actor hacia el sistema. La mayoría de los requerimientos funcionales, sino todos, se pueden expresar con casos de uso.

   Según el autor Sommerville, los casos de uso son una técnica que se basan en escenarios para la obtención de requerimientos. Actualmente, se han convertido en una característica fundamental de la notación UML (Lenguaje de modelado unificado), que se utiliza para describir modelos de sistemas orientados a objetos.

6. **RequisitePro:** es la herramienta que ofrece Rational Software para tener un mayor control sobre los requerimientos planteados por el usuario y los que surjan durante el ciclo de vida del proyecto.

   Se integra con aplicaciones para la administración de cambios, herramientas de modelado de sistemas y con herramientas de pruebas. Esta integración asegura que los diseñadores conozcan los requerimientos del usuario, del sistema y del software en el momento de su desarrollo.

   El desarrollo de software es una tarea de equipo, de tal forma, es importante que todos los miembros del equipo posean un entendimiento compartido de la visión de sus proyectos, metas, especificaciones y requerimientos.

   De igual forma permite al grupo crear y compartir sus requerimientos utilizando métodos familiares basados en documentos potenciados por la aplicación de las capacidades de una base de datos, tales como la trazabilidad y análisis de impacto. El resultado es una mejor comunicación y administración de requerimientos con una mayor probabilidad de completar los proyectos en tiempo, dentro del presupuesto y superando las expectativas.

   Sus ventajas son:

   - Un producto potente y fácil de utilizar para la gestión de requisitos y casos de uso que propicia una óptima comunicación, mejoras en el trabajo en equipo y reduce el riesgo de los proyectos.
   - Combina la interfaz conocida y fácil de utilizar de los documentos de Microsoft Word con potentes funciones de base de datos para conseguir la máxima eficacia en análisis y consulta de requisitos.

- Proporciona a los equipos la posibilidad de comprender el impacto de los cambios.
- Garantiza que todos los componentes del equipo estarán informados de los requisitos más actuales para asegurar la coherencia.
- Proporciona acceso basado en Web para los equipos distribuidos. (Arias, 2006, pp. 6 -10)

Casos de uso

Un caso de uso (use case) es una secuencia de acciones realizadas por el sistema que producen un resultado observable y valioso para alguien en particular. Todo sistema ofrece a sus usuarios una serie de servicios. Es justamente una forma de representar como alguien (persona u otro sistema) usa el sistema.

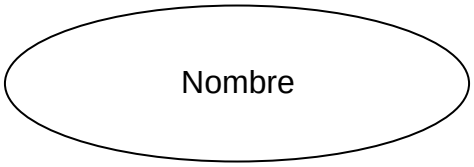
El caso de uso al dar una respuesta a un evento que inicia un agente externo (llamado actor), debe ser desarrollado en función a lo que los usuarios necesitan. Es pues en esencia una interacción típica entre el usuario y el sistema, aunque también puede ser invocado por otro caso de uso.

La idea fundamental en los casos de uso es definir los requerimientos desde el punto de vista de quien usa el sistema y no de quien lo construye. De esta manera, se asegura conocer los requerimientos del usuario para poder construir el software y denotar una operación completa desarrollada por el sistema.

Se debe mencionar que se puede aplicar la técnica de los casos de uso a todo el sistema o a partes, incluyendo a sus subsistemas o a un elemento individual como pueden ser las clases e interfaces.

Representación gráfica de los casos de uso

Los casos de uso se representan mediante elipses en cuyo interior se encuentra su nombre.



Nomenclatura de los casos de uso

Los casos de uso son acciones que debe realizar el sistema, es por ello que debe nombrárseles mediante un verbo seguido por el principal objeto que es afectado por la acción. A veces es conveniente utilizar un prefijo delante del nombre de caso de uso, el cual debe indicar el paquete en el que se ubica (un paquete es un elemento para agrupar a otros), este es llamado path name, mientras que si solo se muestra el nombre del caso de uso se le llamará simple name.

Ejemplos de casos de uso con simple name son: colocar orden, validar usuario, etc., y de casos de uso con path name serán ventas::ingresar pedido, almacén::despachar producto, entre otros, donde ventas y almacén son los nombres dados a los paquetes que agrupan a los casos de uso ingresar pedido y despachar producto respectivamente.

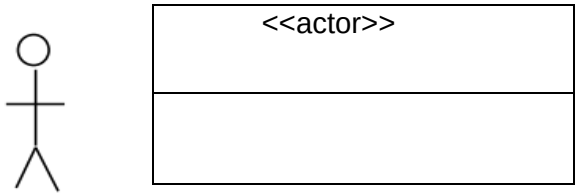
Representación gráfica de un actor

Los actores se representan mediante "hombres de palo" (stick man) tal como se muestra más adelante. Se pueden utilizar los mecanismos de extensión previstos en el UML para estereotipar un actor proveyendo un

icono diferente que pueda ofrecer mejor visibilidad para su propósito. Por ejemplo puede representar un actor mediante un rectángulo con el estereotipo «actor».

Recordar que un actor también es un clasificador y por lo tanto puede representarse como una clase. Cada tipo de actor tiene un conjunto de especificaciones idénticas a la especificación de una clase esto es un conjunto de compartimentos, pero con el campo adicional del estereotipo «actor».

Cuando el actor resulta ser un sistema se suele representar mediante una computadora, para resaltar este hecho.



Posibles representaciones de un actor. El "hombre de palo" es la más utilizada. La última representación es usada solo cuando se trata de sistemas.

Nomenclatura de un actor

Ya que para cada caso de uso pueden existir diversos actores a cada uno de ellos se le tiene que asignar un nombre. El nombre del actor se escribe debajo del icono que lo representa.

Relaciones en los diagramas de casos de uso

Un diagrama muestra las relaciones entre los actores y los casos de uso dentro de un sistema. Estas pueden ser de los siguientes tipos:

- Relaciones de asociación entre actores y casos de uso.
- Relaciones de generalización entre actores.
- Relaciones de generalización entre casos de uso.
- Relaciones incluye (include) entre casos de uso.
- Relaciones extiende (extend) entre casos de uso.

Estas relaciones son fuente de confusión así que preste especial atención a su explicación. En los diagramas una relación de asociación representa la participación de un actor en un caso de uso.

Es la más general de las relaciones y la relación semántica más débil, siempre parte de los actores y viajan en una sola dirección.

A esto también se le conoce como relación de comunicación y suele estereotiparse como <<comunicates>> aunque no es indispensable pues es la única posible entre un actor y un caso de uso.

Representación gráfica de una asociación

Se representa mediante una línea sólida; como siempre parten de los actores hacia los casos de uso no necesita colocarse una cabeza de flecha que indique la dirección.

Relación <<include>>

Al desarrollar un diagrama a menudo se encuentran casos de uso que son incluidos como parte de otro u otros casos, y es que algunos pueden compartir un comportamiento común, que es “factorizado” en versiones de casos de uso especializados.

Una relación «include» significa que el caso de uso base incorpora explícitamente el comportamiento de otro caso de uso.

El caso de uso base siempre utiliza al caso de uso incluido. El objetivo de la relación «include» es permitir invocar el mismo comportamiento muchas veces, poniendo la conducta común que puede ser invocada por otros casos de uso.

De manera general una relación «include», es una relación de dependencia, puesto que su ejecución depende siempre del caso de uso; base, pues es este el que invoca. El caso de uso incluide no puede ejecutarse sin el que lo incluye.

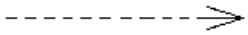
La relación «include» es también un ejemplo de delegación, pues toma un grupo de responsabilidades del sistema y los ubica en un lugar (el caso de uso incluido), el cual es "invocado" por otro caso de uso cuando se necesita usar esa funcionalidad.

Observar que la utilización de la relación «include», está más ligada al concepto de descomposición funcional (muy común en los modelos estructurados) que a los conceptos orientados a objetos. Esto es así, porque los casos de uso solamente sirven para averiguar lo que el usuario necesita del sistema y no cómo lo hace. Cuando se necesite conocer más detalles acerca del caso de uso se recurre a otros tipos de diagramas que estén más relacionados con el enfoque orientado a objetos.

Representación gráfica de una relación «include»

Se representa mediante una línea discontinua con una cabeza de flecha abierta, desde el case de uso base hacia el caso de uso incluido. La dirección de la flecha significa que "el caso de uso base incluye al caso de uso incluido".

«include»



Representación de una relación include

Nomenclatura de una relación «include»

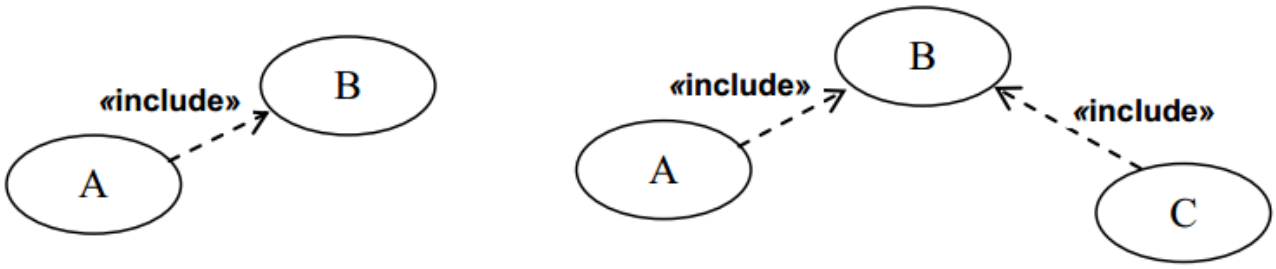
La flecha es nombrada con la palabra clave «include» y no es necesario colocarle algún otro nombre.

Casos típicos

Una relación «include» desde un caso de uso A hacia un caso de uso B, indica que una instancia de A debe también incluir el comportamiento especificado por B.

El comportamiento es incluido junto a la ubicación en la cual está definida A. Esto significa que cada vez que se utilice el caso de uso A, siempre se utilizará el caso de uso B.

Es posible que el caso de uso B, pueda ser invocado por varios casos de uso, tal como se muestra en el diagrama adjunto. Esto indica que B, es un comportamiento común a A y C, que ha sido "factorizado" para evitar definirlo nuevamente, permitiendo su reutilización.



Fuente: Rojas (s.f.)

Relación «extend»

Una relación «extend» significa que se ejecuta el caso de uso base pero bajo ciertas condiciones, es decir, este caso de uso llama a otro que extiende el comportamiento del primero. Esto significa que el caso de caso de uso base implícitamente incorpora el comportamiento del otro.

Se debe utilizar para modelar la parte del caso de uso que tiene un comportamiento opcional.

Para encontrar las relaciones «extend», se deben observar los casos de uso similares, pero que contengan alguna diferencia en cómo realizan las operaciones y qué casos de uso redefinen la forma de efectuar estos procedimientos dentro de otro caso de uso. Se debe pensar en la conducta normal en un caso y la conducta inusual en otro caso, unidos por la relación «extend».

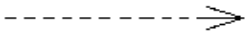
De manera general una relación «extend», es también una relación de dependencia, puesto que el caso de uso extendido entra en acción según

las condiciones que se den al efectuarse el caso de uso base. Recordar que el caso de uso extendido, sólo se utilizará bajo ciertas condiciones.

Representación gráfica de una relación «extend»

Se representa mediante una línea discontinua con una cabeza de flecha abierta, desde el caso de uso extendido hacia el caso de uso base. La dirección de la relación significa que el caso de uso extendido, extiende al caso de uso base.

«extend»



Representación de una relación extend

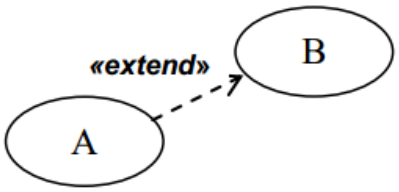
Nomenclatura de una relación «extend»

La flecha es nombrada con el estereotipo «extend». La condición de la extensión es opcionalmente presentada después de la palabra clave.

Casos típicos

Una relación «extend» desde un caso de uso A hacia un caso de uso B indica que una instancia de B puede ser extendida por el comportamiento especificado por A. El caso de uso A, será ejecutado cuando al ser ejecutado B, se den las condiciones que activen a A.

Esta relación está sujeta a las condiciones especificadas en la extensión. El comportamiento es insertado en la ubicación definida en el punto de extensión de B, el cual es referenciado por la relación «extend».

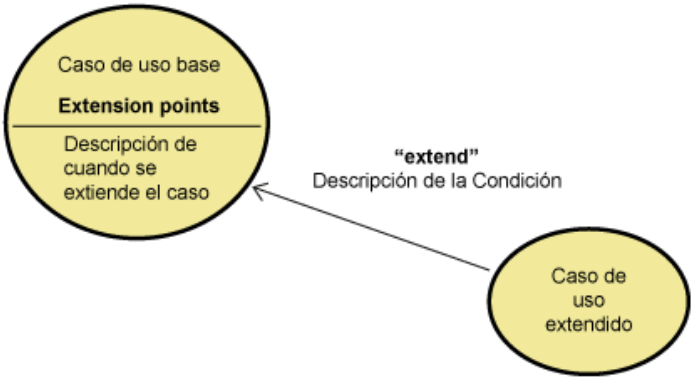


Puntos de extensión en un caso de uso

Una forma de extender la especificación de un caso de uso dentro de la misma elipse que lo representa mediante una cabecera denominada Extensión Point. Un caso de uso puede tener más de un punto de extensión. Dentro de las elipses también se puede mostrar compartimientos presentando atributos, operaciones y por supuesto puntos de extensión.

La descripción de la extensión normalmente se realiza en texto ordinario, pero también se puede especificar en otras formas, como un diagrama de estados o mediante pre o postcondiciones.

El diagrama adjunto muestra la representación de un caso de uso extendido y la especificación de las condiciones para que el caso de uso base sea extendido.



Fuente: Rojas (s.f.)

Cuándo usar «include» y «extend»

Ambos tipos de relación implican la factorización de comportamientos comunes de varios casos de uso; sin embargo, en la relación «include» se trata de factorizar comportamientos comunes para no repetir la definición y los casos de uso factorizados pueden ser utilizados por otros casos de uso; mientras que en la relación «extend» se tienen en cuenta los factores variantes, estos casos que ocurren bajo ciertas circunstancias, poniendo este comportamiento en otro caso de uso extendiendo al caso base. Se deben organizar los casos de uso extrayendo la conducta común mediante relaciones «include» y distinguiendo las variaciones mediante relaciones «extend», con el objetivo de crear un simple, balanceado y comprensible conjunto de casos de uso para su sistema.

En la relación «extend» los actores siguen "conectados" con los casos de uso extendidos. En la relación «include», es el caso de uso base el que se "conecta" con el caso de uso incluido al invocarlo.

Sugerencia:

- Utilizar «extend» cuando se describa una variación de la conducta normal.
- Utilizar «include» cuando un caso de uso siempre es usado por otro u otros casos de uso y se desee evitar repeticiones. (Rojas, s.f. pp. 48-57)

Tipos de casos de uso

A partir de la aplicación de los casos de uso en la práctica, aparecen distintas clasificaciones, que son útiles según el tipo de sistema o el modelo de ciclo de vida utilizado.

Esenciales o de trazo grueso vs de implementación o de trazo fino

Uno de los modelos de ciclo de vida de desarrollo de sistemas que más popularidad ha ganado en los últimos años es el llamado “modelo incremental”, en el cual se van entregando versiones parciales del sistema, que implementan una parte de su funcionalidad. La recomendación en este caso pasa siempre por identificar todos los requerimientos que uno pueda definir sus prioridades y seleccionar cuáles se van a ir implementado en cada versión. Sin embargo, no se pueden especificar en detalle todos los requerimientos: se deben tener apenas el nivel de detalle suficiente para poder definir sus prioridades y comprenderlos en términos generales.

Para aplicar los casos de uso a desarrollos incrementales, se empieza por identificar todos los casos de uso del sistema, sólo al nivel de su nombre. Una vez identificados, se expresan en “trazo grueso”, esto es:

- Ignorar detalles sobre la forma de la interacción entre el actor y el sistema.
- Sólo incluir las alternativas más relevantes, ignorando la mayoría de los errores que aparecen en el uso del sistema.
- No entrar en detalle sobre las acciones que realiza el sistema cuando el usuario interactúa con él. Por ejemplo, si la empresa tuviera una política de descuentos para sus clientes, no es necesario especificar cómo es esa política: alcanza con saber que existe una y que debe ser tenida en cuenta.

De esta forma, se termina con una descripción “gruesa” de todos los casos de uso. Esto sirve para tomar mejores decisiones, junto con los usuarios, sobre qué casos de uso implementar en cada fase. Por otro lado, permite analizar los principales aspectos de todos los casos que afectan al diseño.

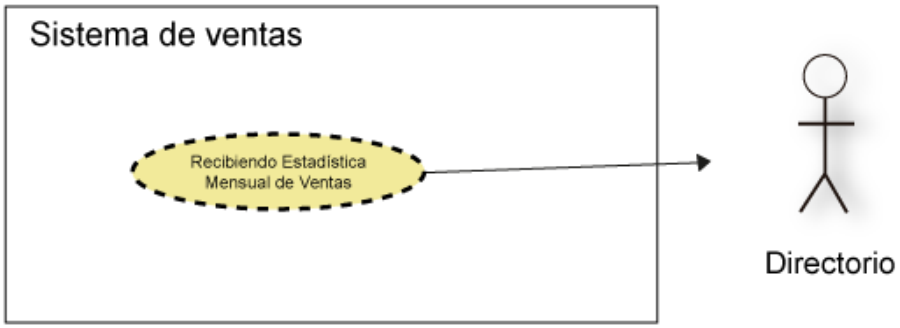
Los casos de uso de trazo fino son aquellos que se especifican una vez que se ha tomado la decisión de implementarlos. En este momento se debe completar todos los detalles que se dejan pasar:

- A medida que se hacen prototipos de las interfaces con los usuarios, se incluyen detalles sobre la forma de la interfaz en la descripción del caso. Por ejemplo, incluir detalles como: “el operador puede en cualquier momento pasar de la ventana de datos del cliente a la ventana de datos del pedido”.
- Si bien esto implica anticiparse al diseño, esto no es negativo, ya que es prácticamente imposible –y perjudicial– hablar con los usuarios siempre en términos de un sistema abstracto.
- Incluir otras alternativas. En particular especificar todos los errores o excepciones que provienen de requerimientos de los usuarios. Para esto se debe tener en cuenta que un sistema tiene dos tipos de errores o excepciones: las que provienen de las definiciones del negocio y las que provienen del procesamiento interno del sistema. Por ejemplo, pensar en un requerimiento del tipo: “si un cliente hace un pedido por un monto mayor al autorizado, se debe rechazar el pedido”. Esta excepción es claramente un requerimiento, y debe ser incluida en las alternativas de los casos de uso. Por el contrario, una excepción del tipo: “Si el usuario ingresa una letra en el lugar del código del producto se le informa que el código de producto debe ser numérico” no debe ser incluida en esta etapa del análisis.
- Especificar con más detalle el comportamiento interno del sistema. El ejemplo de los descuentos, debe especificar cómo es esa política, en un nivel de detalle suficiente para luego poder diseñar una forma de implementarla dentro del sistema.

Casos de uso temporales

Los casos de uso tienen un actor que los inicia y uno o más actores que participan de él. En muchos casos, el inicio de una determinada funcionalidad del sistema es provocado exclusivamente por el paso del tiempo.

Suponer que el sistema de ventas debe generar en forma automática un conjunto de estadísticas para ser entregadas al directorio de la empresa el último día hábil de cada mes. En este caso, el paso del tiempo es el que inicia el caso de uso y el directorio es el actor del sistema. Sin embargo, para expresar claramente que es el paso del tiempo el que inicia el caso, se puede incluir un símbolo representando un reloj en el gráfico de casos de uso o emplear una línea punteada en el borde del óvalo del caso.



Fuente: Ceria (s.f.)

Casos primarios vs. casos secundarios

Jacobson hace referencia a la diferencia entre los casos de uso primarios del sistema y aquellos que no se corresponden con procesos del negocio y cuya ejecución sólo es necesaria para que el sistema funcione normalmente.

Suponer que el sistema requiere de un proceso de depuración de los pedidos que ya han sido cumplidos hace más de 5 años, para evitar que se acumulen indefinidamente. El caso de uso por el cual se depuran estos pedidos, cuyo actor es un administrador del sistema, es considerado un caso secundario, ya que no es central al sistema, sino que es necesario para que el sistema pueda funcionar sin problemas. En la experiencia se ve que no todos los casos de uso secundarios se pueden identificar en la etapa de requerimientos, ya que muchos de ellos dependen de decisiones de implementación que se toman en la etapa de diseño.

El proceso de análisis de requerimientos con casos de uso

Esta sección describe los pasos a seguir para aplicar la técnica de análisis de requerimientos con casos de uso.

Identificar los actores

Si la primera pregunta que un analista debe hacer a sus usuarios es ¿Para qué es este sistema?, la segunda es claramente ¿Para quiénes es este sistema? Como se mencionó al hablar sobre los actores, identificar a todos ellos es crítico para un buen análisis de requerimientos. Por lo tanto, antes de avanzar con los casos de uso, se debe tratar de identificar todos los tipos de usuario diferentes que tiene el sistema. Si este funcionará en una empresa, se debe preguntar cuáles de las áreas afectadas usarán o actualizarán su información.

A pesar de hacer una identificación inicial de los actores, también debe repetirse a medida que se empieza a describir los casos de uso, ya que al conocer más detalles del sistema pueden aparecer nuevos tipos de usuarios.

Identificar los principales casos de uso de cada actor

El siguiente paso es enunciar los nombres de los principales casos de uso de cada uno de los actores que se identificó en el paso anterior. No es necesario especificar cuáles son las acciones dentro del caso de uso. Tampoco se debe preocupar si no aparecen muchos casos, ya que hay técnicas para encontrar nuevos casos de uso a partir de los existentes.

Identificar nuevos casos a partir de los existentes

Uno de los principales errores que se pueden cometer al identificar requerimientos es olvidarse de alguno. Como los estos están en la cabeza de los usuarios, el éxito de esta tarea depende de la habilidad del programador. Una ayuda para identificar nuevos casos de uso a partir de los casos existentes, es aplicar las mismas técnicas utilizadas para identificar eventos según el análisis estructurado. Es decir que se basa en el análisis de cuatro situaciones posibles a partir de los requerimientos ya identificados.

Casos de uso “opuestos”

Hay dos formas de buscar casos de uso opuestos. La primera es buscar la función puesta a la descrita por el caso. Por ejemplo, en el caso de realizar un pedido, podemos pensar que la función opuesta es cancelar ese mismo pedido. Si cancelar pedidos es una función que mi sistema debe realizar, y que no había identificado anteriormente, acabo de evitarme un error que puede ser muy costoso de corregir más adelante.

La otra forma de buscar casos de uso opuestos es pensar en la negación de la acción principal del caso de uso. Supongamos que tenemos un caso de uso “Pagando Pedido”, que es realizado por el actor cliente. El caso “No

Pagando Pedido", puede de nuevo ser significativo. Si el cliente no paga el pedido, tal vez nuestro sistema deba hacer algo, y podemos estar frente a un nuevo caso de uso.

Casos de uso que preceden a casos existentes

Una buena pregunta para hacer es:

¿Qué es lo que tiene que ocurrir antes de este caso de uso?

En el caso del cliente que hace un pedido, son muchas las cosas que pueden ocurrir antes de ese caso:

1. El cliente, por ejemplo, debe ser cliente. En esta situación tal vez tenga un nuevo caso de uso "Ingresando Cliente", que puede o no ser un caso del sistema.
2. El cliente debe poder consultar cuáles son los productos existentes. Probablemente este sea un caso de uso que ya haya sido identificado. Sin embargo, usando esta técnica muchas veces se descubren nuevos requerimientos.

Casos de uso que suceden a casos existentes

Esto es similar al punto anterior. La pregunta que se debe hacer es: ¿Qué ocurre después de este caso de uso?

En el ejemplo de los pedidos, es evidente que la mayoría de la funcionalidad del sistema recién empieza cuando el cliente hace un pedido. Por lo tanto, analizar ¿cómo sigue la historia? es una buena forma de asegurar que no se están dejando requerimientos sin identificar.

Crear descripciones de casos de uso de trazo grueso

Una vez identificados todos los casos de uso, se empiezan a documentar sus pasos. Esta tarea no es estrictamente secuencial de la anterior: es posible seguir buscando otros nuevos.

La documentación de los casos de uso identificados debe hacerse del tipo "trazo grueso", salvo que deba implementarse en la primera iteración. (Ceria, s.f. pp. 11-15)

Pasos para la definición de un caso de uso

Para realizar un completo y minucioso caso de uso y poder suministrar a los programadores, analistas y clientes la información detallada sobre el sistema se deben tener en cuenta los siguientes datos:

- Id.
- Nombre.
- Referencias cruzadas.
- Creado por.
- Última actualización por.
- Fecha de creación.
- Fecha de última actualización.
- Actores.
- Descripción.
- Trigger.
- Pre-condición.
- Post-condición.
- Flujo normal.
- Flujos alternativos.
- Includes.
- Frecuencia de uso.
- Reglas de negocio.
- Requerimientos especiales.
- Notas y asunto.

Los casos de uso tienen las siguientes características:

1. Están expresados desde el punto de vista del actor.
2. Se documentan con texto informal.
3. Describen tanto lo que hace el actor como lo que hace el sistema cuando interactúa con él, aunque el énfasis está puesto en la interacción.
4. Son iniciados por un único actor.
5. Están acotados al uso de una determinada funcionalidad –claramente diferenciada– del sistema.

El último punto es tal vez el más difícil de definir. Se puede decir que todo sistema tiene un único caso de uso empleando el sistema. Sin embargo, la especificación resultante sería de poca utilidad para entenderlo; estaría como implementando un gran sistema escribiendo un único programa.

La pregunta importante es: ¿Qué es una funcionalidad claramente diferenciada? Por ejemplo, ¿ingresar pedidos es un caso de uso y autorizarlos es otro? ¿Cancelar los pedidos, es otro caso de uso o es parte del referido al ingreso de pedidos? Si bien se pueden encontrar argumentos válidos para cualquiera de las dos alternativas, en principio la respuesta a todas estas preguntas es que son todos casos de uso distintos.

Lamentablemente, si en la programación los criterios para dividir la funcionalidad en programas suelen ser difusos, los criterios para dividir la funcionalidad de un sistema en casos de uso son aún más complicados y por esto se hace importante usar el sentido común en estas decisiones.

En principio se puede decir que la regla general es: una función del sistema es un caso de uso si se debe indicar explícitamente que se quiere acceder a esa función. Por ejemplo, si se desea dar de alta un pedido, accederá a la funcionalidad de alta de pedidos del sistema. Sin embargo, si se pretende dar de alta un campo del pedido, no debe indicar al sistema que intenta

acceder a esa función. Dar de alta un campo de un pedido es una función que forma parte de un caso de uso mayor: dar de alta un pedido.

Esta regla, si bien puede ser útil, no debe seguirse al pie de la letra, ya que se puede prestar a confusiones. Por ejemplo, si se quiere especificar un sistema en el cual los usuarios pueden ver un pedido y tienen disponibles funciones para ver el siguiente pedido, el anterior, el último y el primero. El actor debe indicar al sistema que desea acceder a cada una de esas funciones y según la regla serían todas ellas casos de uso distintos. Sin embargo, en esta situación es mucho más práctico definir un único caso de uso navegando pedidos, que especificarlos todos como casos de uso distintos.

Cuando se piensa en el grado de detalle de la división de los casos de uso también resulta útil imaginar que se está escribiendo el manual del usuario del sistema. A nadie se le ocurriría escribir un manual de usuario con un solo capítulo en el que se describe toda su funcionalidad. De la misma forma, no se debe hacer una especificación con un solo caso de uso. (Ceria, s.f. pp. 5-6)

Referencias

- Arias, M. (2006). *La ingeniería de requerimientos y su importancia en el desarrollo de proyectos de software.* Consultado el 28 de enero de 2014*, en* revistas.ucr.ac.cr/index.php/intersedes/article/download/790/851

- Ceria, S. (s.f.). *Casos de uso, un método práctico para explorar requerimientos*. Consultado el 28 de enero de 2014, en http://www-2.dc.uba.ar/materias/isoft1/2001_2/apuntes/CasosDeUso.pdf

- Rojas, R. (s.f.). *Gestión informática I*. Consultado el 28 de enero de 2014, en http://www.cacvirtual.upla.edu.pe/distancia/as_cf.php/05/Gestion%20Informatica%20I.pdf

- Servicio Nacional de Aprendizaje, SENA. (2010). *Diseño de casos de uso*. Colombia: Autor.

- Sommerville, I. (2005). *Ingeniería del Software*. México D.F. Editorial Pearson.

Control del documento

	Nombre	Cargo	Dependencia	Fecha
Autor	Nidyan Slendy Mantilla Daza	Contratista	Centro Agroturístico Regional Santander	Diciembre de 2013
Adaptación	Ana María Mora Jaramillo	Guionista - Línea de producción	Centro Agroindustrial Regional Quindío	Enero de 2014