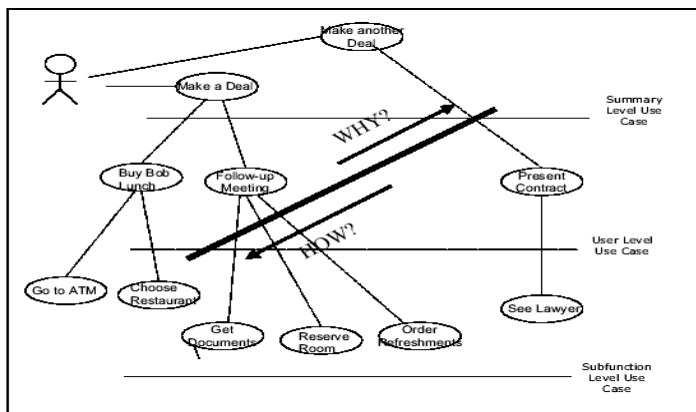


Writing effective use cases

Alistair Cockburn



Maggi, Germán
Palmitesta, Mauro

Indice

1. Actores	2
1.1 En el comienzo de la producción del caso de uso	2
1.2 Durante la descripción del caso de uso, y durante el diseño	3
1.3 Después del diseño, preparando la implementación del sistema	4
1.4 Caracterizando los actores primarios	5
2. Alcance	6
2.1 Alcance funcional	6
2.1.1 Lista Actor-Meta	6
2.1.2 Resúmenes de Caso de Uso	7
2.2 Alcance de Diseño	7
2.3 Usando iconos gráficos para el alcance de diseño	8
3. Tres Niveles de Metas	10
3.1 Meta de Usuario	10
3.2 Meta de Resumen	12
3.2.1 Casos de Usos de gran altura	13
3.2 Subfunciones	13
3.3 Conclusión	14
3.4 Encontrando el nivel correcto	14
3.4.1 Meta de Usuario	14
3.4.2 Longitud del caso de uso	15
4.Precondiciones, Disparadores, Garantías	16
4.1 Precondiciones	16
4.2 Garantías mínimas	17
4.3 Garantías de éxito	18
4.4 Disparadores	18
5.Escenarios y Pasos	20
5.1 El escenario principal, escenarios	20
5.1.1 El escenario principal como el caso simple	20
5.1.2 Estructura común	20
5.1.3 El cuerpo del escenario	21
5.2 Pasos	21
5.2.1 Directrices para los pasos	22
6.Extensiones	27
6.1 Condiciones de extensión	28
6.1.1 Piensa todos los fallos imaginables y cursos alternativos....	28
6.1.2 Racionalizar la lista de extensiones	29
6.1.3 Combinar fallos	30
6.2 Manejo de extensiones	30
6.2.1 Fallos dentro de fallos	32
6.2.1 Creando un nuevo caso de uso de una extensión	32

1. El actor primario de un caso de uso

El actor primario es la persona interesada en el funcionamiento del sistema que usa el mismo para obtener uno de sus servicios. El actor primario tiene una meta respecto al sistema que puede ser satisfecha mediante su operación. El actor primario es usualmente, pero no siempre, el actor que dispara el caso de uso. Generalmente, el caso de uso comienza porque el actor primario envía un mensaje, presiona un botón, teclea algo, o de alguna otra forma inicia la historia. Hay dos situaciones comunes en que el iniciador del caso de uso no es el actor primario. La primera es cuando un empleado de la empresa u operador telefónico inicia el caso de uso en nombre de la persona a la cual realmente le interesa, la segunda es cuando el caso de uso es iniciado por el tiempo.

Un empleado de la empresa u operador telefónico es generalmente una conveniencia tecnológica para la persona realmente interesada, a la cual llamaremos *actor primario ulterior*. Con un cambio de tecnología es muy probable que el actor primario ulterior sea el iniciador directo del caso de uso, usando la web o un sistema telefónico automático. Un ejemplo de esto es un cliente que actualmente hace un pedido telefónico. En un rediseño para la web del sistema, el usuario puede ingresar su pedido directamente, como en Amazon.com.

Similarmente, las divisiones de Marketing o Auditoría pueden insistir en la presencia de casos de uso que son operados por un empleado. En realidad no es la meta de los empleados lo que hace ejecutar el caso de uso, ellos son una conveniencia tecnológica para los gerentes de Marketing. Bajo circunstancias ligeramente diferentes, los gerentes de marketing podrían ellos mismos ejecutar los casos de uso.

Escribimos “reporte de ventas para el cliente” o “el empleado...para el Departamento de Marketing” para capturar que el usuario del sistema está actuando para alguien más. Esto nos permite conocer que la interfaz de usuario y los permisos de seguridad necesitan ser diseñados para el empleado, pero el cliente o el Departamento de Marketing son los realmente interesados en el resultado.

El tiempo es el otro ejemplo de un actor iniciador que no es el operador. No hay empleados disparando los casos de uso que corren cada medianoche, o al final del mes. Es fácil, en este caso, ver que el actor primario es cualquier persona que tenga interés de que el caso de uso se corra en ese momento.

Es fácil involucrarse en largas discusiones en la cuestión de usuarios versus actor primario ulterior. Sugerimos que no gaste demasiado tiempo en ello. Cuando el equipo empieza a investigar el diseño de las interfaces del usuario, ellos pondrán, o deberían poner, gran energía en estudiar las características de los usuarios reales. Cuando ellos revelan los requerimientos, encontrarán útil saber el actor primario ulterior de cada caso de uso, quien es el que realmente importa.

Un estudiante nos preguntó astutamente “¿Cuánto daño hay si obtengo mal el actor primario en este punto?. La respuesta es “No mucho”.

¿Por qué el actor primario es poco importante e importante a la vez? Los actores primarios son importantes en el comienzo de la etapa de requerimientos, y justo antes de la entrega del sistema. Entre esos dos puntos, se vuelven poco importantes.

1.1 En el comienzo de la producción del caso de uso

Listar los actores primarios nos ayudan a tener en mente el sistema entero por un breve momento. Por este motivo proponemos nombres de actores con el propósito de nombrar todas las metas propuestas, ya que son las metas las que realmente nos interesan. El motivo por el cual no proponemos las metas directamente, es que de esta manera se pierden muchas de ellas. Proponer los actores primarios establece una estructura de trabajo. Podremos entonces recorrer esa estructura para obtener la lista de metas.

Crear un gran número de actores primarios no nos perjudica, ya que en el peor de los casos, generamos la misma meta dos veces. Cuando inspeccionamos los actores y las metas para priorizar el trabajo, encontraremos y eliminaremos los duplicados.

Incluso con este proceso de proponer en dos pasos (actores y metas), es bastante improbable que propongamos todas las metas que nuestro sistema necesita soportar. Algunas tienden a descubrirse mientras se escriben los pasos para el manejo de fallos del caso de uso. Sin embargo, esto es algo que no nos debe afectar en esta etapa temprana, por lo que haremos lo mejor para capturar todas las metas listando primero todos los actores primarios.

Una rica lista de primeros actores confiere tres otras ventajas:

- Enfoca nuestra mente en las personas que usarán el sistema. En el documento de requerimientos, escribimos quien esperamos que sea el actor primario, las descripciones de su trabajo, su típico ambiente y aptitudes. Al hacer esto los diseñadores de las interfaces de usuario y del sistema pueden hacer corresponder éstos con esa habilidad.
- Establece la estructura para la lista actor-meta, que será usada para priorizar y particionar el trabajo de desarrollo.
- Podrá ser usado como partición de un gran conjunto de casos de uso en paquetes.

1.2 Durante la descripción del caso de uso, y durante el diseño

Una vez que empezamos a desarrollar el caso de uso en detalle, el actor primario se convierte en poco importante. Esto puede sorprenderlo. ¿Qué ocurre si sobre la marcha, los escritores de casos de uso descubren que un caso de uso puede ser usado por varias clases de actores? Por ejemplo, alguien con más jerarquía que el empleado puede responder el teléfono y hablar con el cliente. Los escritores de casos de uso usualmente comienzan nombrando progresivamente el actor primario en una forma genérica, usando roles como *Receptor de siniestros*, *Receptor de órdenes*, *productor de factura*, y así. Esto produce que los casos de uso digan “El productor de facturas produce la factura...”, y “El receptor de órdenes recibe la orden..”.

Se puede manejar esto de varias maneras, cada una con pequeñas ventajas y desventajas. Ninguna estrategia es claramente superior, sólo se debe escoger una:

- **Alternativa 1:** Divida el actor primario en los roles que el juega manteniendo una tabla que liste todas las diferentes clases de personas y sistemas que son actores primarios de algún caso de uso, y todos los roles que ellos pueden jugar. Use el nombre del rol en el campo del actor primario y la lista actor-rol para obtener de los casos de uso las personas y los sistemas del mundo real. Esta estrategia permite a los escritores ignorar el problema de dar títulos a los roles y simplemente tratar con la escritura del caso de uso. Alguien, quizás el diseñador de interfaz de usuario, usará la lista actor-rol para corresponder el caso de uso con los eventuales usuarios. El problema de la alternativa 1 es que hay una lista aparte para mantener y leer.
- **Alternativa 2:** Escribir, en algún lugar en el frente del caso de uso “El Gerente puede usar cualquier caso de uso que puede usar el empleado, y algo más. El Gerente Regional puede usar cualquier caso de uso que puede usar el Gerente, y algo más. Por lo tanto, donde escribamos que el actor primario es el empleado, se sobreentiende que cualquier persona con mayor jerarquía, el Gerente y el Gerente Regional en este caso, pueden también usar el caso de uso”. Esto es más fácil de mantener que la tabla actor-rol, ya que es improbable que cambie. La desventaja es que las personas gastarán más tiempo recordando que cuando el empleado es el actor primario, en realidad el Gerente también puede usar el caso de uso.

Se alcanzan buenos resultados con ambos métodos. Prefirimos usar la segunda alternativa ya que tenemos una tabla menos que escribir, revisar y mantener.

El punto es que el campo Actor Primario en la plantilla del caso de uso se devalúa al pasar el tiempo. Esto es normal, y no debería preocuparnos.

1.3 Después del diseño, preparando la implementación del sistema

Justo antes de entregar el sistema, el actor primario se vuelve importante otra vez. Necesitamos la lista de todas las personas, y cuáles casos de uso ellos usan. Los necesitaremos para:

- Empaquetar el sistema en unidades que se cargarán en varias máquinas de usuario.
- Establecer niveles de seguridad para cada caso de uso (usuarios web, internos, supervisores, etc.).
- Crear la capacitación para los distintos grupos de usuarios.

En conclusión, se debe concentrar en la integridad y exactitud de la lista de actores primarios desde el comienzo. **El daño de omitir un actor es grande:** podemos pasar por alto una sección entera de requerimientos.

El daño de encontrar demasiados actores es pequeño: algo de trabajo extra en las etapas tempranas hasta que los actores innecesarios sean eliminados. Después de eso, los actores se vuelven en realidad poco importante. Sólo antes de la implementación se vuelven otra vez importantes, para la preparación del paquete y la capacitación.

Nota: Actores vs. Roles

La palabra *actor* implica una acción *individual*, pero a veces también queremos denotar una categoría general de individuos que pueden jugar ese rol. Supongamos que Kim es una cliente de MiTel SA, Chris es un empleado, y Pat es Gerente de ventas. Cualquiera de ellos puede *Hacer un pedido*. Usando el lenguaje de *actores*, decimos que Kim, Chris y Pat pueden ser *actores primarios* para el caso de uso *Hacer un pedido*. También decimos que *Cliente*, *Empleado*, y *Gerente de Ventas*, son los actores permitidos para *Hacer un pedido*. Podemos escribir que “un gerente de ventas puede usar cualquier caso de uso que también pueda un empleado”. Son todas buenas formas de decirlo.

Usando el lenguaje de *roles*, decimos que Kim, Chris y Pat son actores individuales. Cualquiera de ellos puede jugar el rol de *Cliente*, pero sólo Chris y Pat pueden jugar el rol de *Empleado*, y sólo Pat puede jugar el rol de *Gerente de Ventas*. Entonces decimos que el rol que conduce el caso de uso *Hacer un pedido* es *Recepcionista de Pedido*, y cualquiera de los tres puede jugar ese rol. Esta forma de describir es más precisa que la previa, y muchos la prefieren.

Nota: Especialización de actores en diagramas UML

UML provee un flecha con punta vacía para indicar que un actor *especializa* a otro. Esa flecha resuelve parte, pero no toda la controversia actor-rol.

Lo bueno de la flecha es que permite expresar en forma concisa que el Gerente puede hacer cualquier cosa que el empleado pueda. Lo malo es que muchas personas lo piensan al revés. No piensan al Gerente como una *clase especial* de Empleado, o un Empleado como una *clase especial* de Cliente, que es lo que los diagramas pretenden representar. Ellos piensan que el Gerente *es más que* un Empleado. Esto no es un gran problema, pero se tendrá que lidiar con estas reacciones.

La especialización no ayuda del todo en la cuestión actor-rol. Un *Empleado de ventas* y un *Empleado de auditoría* tiene casos de uso superpuestos. No se puede usar la especialización para indicar esta superposición, ya que ninguno puede hacer todo lo que el otro puede.

1.4 Caracterizando los actores primarios

Tener sólo una lista de actores no ayuda mucho a los diseñadores. Ellos deberían conocer que clases de habilidades tendrán los actores, entonces diseñan el comportamiento del sistema y las interfaces en base a eso. Los equipos que crean un Mapa del Perfil del Actor mantienen una mejor visión de cómo su software debe satisfacer las necesidades de los usuarios. Esto es porque ellos piensan en las habilidades básicas de sus usuarios finales durante el diseño.

El Mapa del Perfil simplificado del Actor tiene sólo dos columnas, como en el siguiente ejemplo:

Mapa del Perfil Simplificado del Actor:

Nombre	Perfil: Ambiente y Habilidades
Cliente	Persona de la calle, capacitado para usar pantalla sensible, pero no se espera que opere un IGU con facilidad. Puede tener dificultades de lectura, corto de vista o daltónico.
Empleado de mercaderías devueltas	Persona trabajando con software continuamente. Pantalla sensible, usuario sofisticado Puede querer personalizar su interfaz
Gerente – usuario ocasional	Usa interfaz gráfica pero no está familiarizado con ninguna función de software en particular. Impaciente.

2. Alcance

“*Alcance*” es la palabra que usamos para definir la extensión de lo que consideramos que debemos desarrollar.

Seguir el alcance del proyecto, o incluso sólo el alcance de una discusión puede ser difícil. Una buena herramienta para manejar discusiones sobre alcance es la **Lista in/out**, ya que es extremadamente simple y efectiva. Esta puede ser usada para controlar el alcance de las discusiones en reuniones ordinarias así como también el alcance de los requerimientos del proyecto.

Simplemente debemos construir una tabla con tres columnas, la primera con el tópico, y las dos siguientes con **IN** y **OUT** para denotar si está dentro o fuera del alcance. Cada vez que haya una confusión sobre si algún tópico está dentro del alcance, se agrega a la tabla y se consulta a las correspondientes personas si está **IN** o **OUT**. Lo maravilloso es que aunque es completamente claro para cada persona en la habitación si un tópico está **IN** o **OUT**, ellos usualmente tienen visiones opuestas. Aquí tenemos un ejemplo de lista **IN/OUT**:

Tópico	IN	OUT
Facturar de alguna forma		X
Producir reportes sobre pedidos, por ejemplo por vendedor, por parte, etc.	X	
Combinar pedidos en uno	X	
Entregas parciales, entregas demoradas, entregas erróneas	X	
Todo servicio nuevo de sistema software	X	
Cualquier parte del sistema que no sea software		X
Identificación de algún software preexistente que puede ser usado	X	
Pedidos	X	

2.1 Alcance Funcional

El alcance funcional se refiere a los servicios que su sistema ofrece. Este puede ser eventualmente capturado por los casos de uso. Sin embargo, al comenzar un proyecto, es probable que no se conozca precisamente el alcance funcional. El alcance funcional se definirá al mismo tiempo que se identifican los casos de uso. Ambas tareas están entrelazadas. La lista **IN/OUT** ayuda con esto ya que le permite definir el límite de lo que está dentro y lo que está fuera del alcance. Las otras dos herramientas son la **Lista Actor-Meta** y los **Resúmenes de casos de uso**.

2.1.1 Lista Actor-Meta

La **lista Actor-Meta** nombra todas las metas del usuario que el sistema soporta, mostrando el contenido funcional del sistema.

La **lista Actor-Meta** contrariamente a la **lista IN/OUT**, que muestra ítems que están dentro y fuera del alcance, incluye sólo los servicios que actualmente van a ser brindados por el sistema. Para hacer esta lista, se debe construir una tabla de tres columnas. En la primera columna va el nombre del primer actor, en la segunda la meta del actor, y en la tercera la prioridad. Aquí tenemos un ejemplo de la **lista Actor-Meta**:

Actor	Meta	Prioridad
Alguien	Revisar pedidos	1
Supervisor	Cambiar autorizaciones	2
Comprador	Cambiar contacto del proveedor	3
Cliente	Iniciar pedido	1

La **lista Actor-Meta** es el punto inicial de negociación entre el usuario, el financista del proyecto y el grupo de desarrollo. Se concentra en el plan y contenido del proyecto.

El Diagrama principal de casos de uso es una **lista Actor-Meta** en forma visual. Es un hecho que a las personas les agrada la forma en que muestra la agrupación de los casos de uso por actores primarios. Actúa como

un diagrama de contexto y una tabla gráfica de contenidos enlazada a la descripción del caso de uso. A pesar de que provee estos dos propósitos, el diagrama no brinda una hoja de trabajo para estudiar los datos de estimación del proyecto. Si usa el diagrama, necesitará recoger la prioridad y la información de estimación y luego construir una estructura de trabajo.

Para que el diagrama cumpla su principal propósito manténgalo ordenado, mostrando sólo casos de uso con nivel de meta de usuario o mayor.

2.1.2 Resúmenes de caso de uso

Destacamos la importancia de administrar su energía trabajando con los niveles de precisión más bajos tanto como sea posible. La **lista Actor-Meta** es el nivel de precisión más bajo para describir el comportamiento del sistema. Esta es muy útil para trabajar con la imagen total del sistema. El siguiente nivel de precisión será el escenario principal o **Resumen de caso de uso**.

Los **resúmenes de caso de uso** son una descripción de 2-6 oraciones del comportamiento del sistema, mencionando solamente la actividad más significativa y los fallos. Los **resúmenes de casos de uso** ayudan a recordar a las personas de lo que pasa en el caso de uso y son útiles para estimar la complejidad del trabajo.

En algunos proyectos, en los que hay buena comunicación interna y una continua discusión con los usuarios, nunca se escriben más que estos **resúmenes de caso de uso** para capturar requerimientos. Ellos dejan el resto de los requerimientos para las continuas discusiones, prototipos, y frecuentes incrementos.

Se puede representar el **resumen de caso de uso** en una tabla, que es una extensión de la **lista Actor-Meta**, o directamente en el cuerpo del caso de uso como su primer bosquejo.

Actor	Meta	Resumen
Staff de Producción	Preparar fuentes de datos digital cartográfica	El staff de producción convierte datos digitales externos a un formato estándar, validando y corrigiéndola para combinarla con una base de datos operacional. Los datos son catalogados y almacenados en una librería de fuentes digitales.

2.2 Alcance de diseño

El alcance de diseño es la amplitud del sistema. Podría decirse “la amplitud espacial” si el software ocupara espacio. Es el conjunto de sistemas, hardware y software, que están bajo diseño o discusión. Si estamos diseñando un cajero automático, debemos producir hardware y software que irán en el cajero. Lo que debemos diseñar es el cajero y todo lo que hay en él. La red con la que debe comunicarse no está bajo nuestro alcance de diseño.

De ahora en más, cuando se escriba *alcance* solamente, me referiré al **alcance de diseño**. Esto es porque el alcance funcional está adecuadamente definido por la **lista Actor-Meta** y los casos de uso, mientras que el alcance de diseño es un asunto de interés en cada caso de uso.

Es increíblemente importante que el escritor y el lector estén de acuerdo sobre el **alcance de diseño** de un caso de uso. El costo de equivocarse aquí puede multiplicarse con desastrosos resultados para el contrato. Los lectores de un caso de uso deben rápidamente ver lo que se entiende que está dentro del límite del sistema. Esto no será evidente sólo con el nombre del caso de uso o el actor primario. Los sistemas de diferentes tamaños lo presentan incluso dentro del mismo caso de uso.



Usualmente, el escritor de caso de uso considera el **alcance de diseño** del sistema como algo obvio, tan obvio que no lo mencionan. Sin embargo, cuando existen muchos escritores y lectores, el **alcance de diseño** no es tan obvio. Un escritor puede pensar a toda la organización como **alcance de diseño**, otro en todos los sistemas de software de la organización, otro en el nuevo sistema cliente/servidor, y otro sólo en el cliente.

Los lectores, sin indicios sobre lo que se trata, se pierden o mal interpretan el documento.

Una solución es etiquetar cada caso de uso con su alcance de diseño. Los nombres de alcance de diseño son:






- **Alcance de Negocio** 🏠: significa que se está discutiendo el comportamiento de toda la organización para llegar a la meta del actor primario. Se puede utilizar el mismo nombre de la empresa para nombrar

el alcance. Si se trata de un departamento de la misma, se puede utilizar el nombre del mismo. Los casos de uso del negocio están escritos con un alcance de Negocio.

- **Alcance de sistema** : Se utiliza el nombre del sistema para etiquetar el alcance. Significa justamente la pieza de hardware o software del que está encargado de construir. Fuera del sistema están todas las partes de hardware, software y personas con las que se debe interactuar.
- **Alcance de subsistema** : Se utiliza el nombre del subsistema para etiquetar el alcance. Significa que se ha abierto el sistema principal y se tratará como sus partes trabajan.

2.3 Usando iconos gráficos para resaltar el alcance de diseño

Se puede adjuntar un gráfico a la izquierda del título del caso de uso, para señalar el alcance de diseño al lector antes de empezar a leer. Se pueden utilizar los siguientes iconos para representar el alcance de diseño cada caso de uso:

- Un caso de uso del negocio tiene a la organización como alcance de diseño. Su icono es un casita de color gris  si se trata de toda la organización como una caja negra, o blanco  si hace referencia a un departamento o staff dentro de la misma.
- Un caso de uso del sistema tiene un alcance de sistema. Su icono es una caja color gris  si se la trata en su totalidad como caja negra, o blanco  si se hace referencia de cómo trabajan sus componentes.
- Un caso de uso de componente tiene un alcance de subsistema. Su icono es un tornillo 



Ejemplos

Trabajamos para la Compañía telefónica MiTel S.A., la cual está diseñando un nuevo sistema, Acura, para tomar las órdenes de servicio y actualizaciones. Acura consiste de una estación conectada a un servidor. El servidor estará conectado a un mainframe corriendo el viejo sistema, BSSO, que es una terminal conectada al mainframe. No estamos autorizados a hacer cambios a BSSO. Solamente usamos sus interfaces ya existentes.


Los actores primarios de acura incluyen a el cliente, el recepcionista, algunos gerentes, y el sistema BSSO (estamos seguros que BSSO no está en nuestro alcance de diseño).

Vamos a encontrar algunas de las metas que el sistema debe soportar. El más obvio es “Agregar un nuevo servicio”. Decidimos que el actor primario es el recepcionista, actuando con el cliente.

La primera pregunta a hacerse es “¿Qué es el sistema bajo discusión?”. De esto deducimos que hay dos cosas que nos interesan:

- MiTel S.A. Estamos interesados en la pregunta “¿Cómo es visto el servicio de MiTel S.A. por los clientes, mostrando la nueva implementación del servicio completamente, desde la demanda inicial hasta la implementación?” Esta pregunta es de doble interés. Los gerentes de la Compañía querrán ver cómo el nuevo sistema aparece a la vista del mundo exterior, y el equipo de implementación querrá el contexto en el cual estará el nuevo sistema. Este caso de uso será escrito con alcance de negocio , con el campo de alcance etiquetado MiTel SA, y sin hacer mención de los jugadores internos de la compañía (recepcionista, departamentos, etc.) Esta clase de caso de uso es usualmente llamado caso de uso del negocio, ya que trata sobre el negocio en sí.
- Acura. Estamos interesados en la pregunta “¿Cómo se ve el servicio de Acura, en sus interfaces con el recepcionista o el cliente por un lado, y con el sistema BSSO en el otro?”. Este es el caso de uso en el que los diseñadores se interesarán más, ya que es exactamente lo que deben construir. El caso de uso será escrito con alcance de sistema , con el campo de alcance etiquetado como Acura. Ellos mencionarán libremente el recepcionista, departamentos y otras computadoras, pero no harán mención de los subsistemas de la estación y del servidor.

Produciremos entonces dos casos de uso. Para evitar repetir dos veces la misma información, escribimos el caso de uso con alcance de Negocio en un nivel más alto (barrilete), mostrando a MiTel SA respondiendo a la demanda, cumpliéndola o haciéndole algún cambio, y cobrándola. El propósito de caso de uso con alcance de negocio es mostrar el contexto que rodea al nuevo sistema. Entonces describiremos el manejo en 5-20 minutos de las demandas en detalle en el caso de uso con meta de usuario teniendo a Acura como alcance de diseño.



Caso de uso:  **Agregar un nuevo servicio (Negocio)** 

Actor primario: Cliente

Alcance: MiTel SA

Nivel: Resumen

1. El cliente llama a MiTel SA, demandando un nuevo servicio...
2. MiTel SA cumple con...

Caso de uso:  **Agregar un nuevo servicio (Acura)** 

Actor primario: Recepcionista

Alcance: Acura

Nivel: Usuario

1. El cliente llama, y plantea la demanda al recepcionista.
2. El recepcionista encuentra el cliente en Acura.
3. Acura presenta el actual paquete de servicios del cliente.....

No se escribirán los casos de uso con alcance Estación Acura o Servidor Acura, ya que ellos no son de interés para nosotros en este momento. Más tarde, alguien en el equipo de diseño puede decidirse por documentar el diseño del subsistema de Acura usando casos de uso. En ese momento, ellos tendrían que escribir dos casos de uso, uno con el alcance Estación Acura, y otro con alcance Servidor Acura. Mi experiencia es que usualmente estos casos de uso nunca son escritos, ya que hay otras técnicas más adecuadas para documentar la arquitectura de los subsistemas.

3.Tres Niveles de Metas

Hemos visto que ambos, las metas y las interacciones en un escenario pueden ser descompuestas en metas e interacciones de mayor detalle. Esto es normal, y lo vemos cotidianamente en nuestras acciones. Los siguientes párrafos ilustran como nuestras metas contienen sub-metas y sub-metas de sub-metas.

“Quiero realizar esta venta. Para hacer esto debo llevar a comer al encargado compra. Para hacer esto tengo que retirar dinero del cajero automático. Para hacer esto debo obtener la aprobación de la solicitud de identificación del cajero. Para hacer esto debo hacer que se lea mi tarjeta bancaria. Para hacer esto debo buscar la ranura donde insertar la tarjeta bancaria.”

El analista se encontrará con la pregunta en cada oración ¿Cuál es el nivel de meta que debemos describir? Dar nombres a los niveles de las metas puede ayudarnos a responder esta pregunta. Las siguientes secciones describen el nombre de los niveles y sus respectivos iconos.

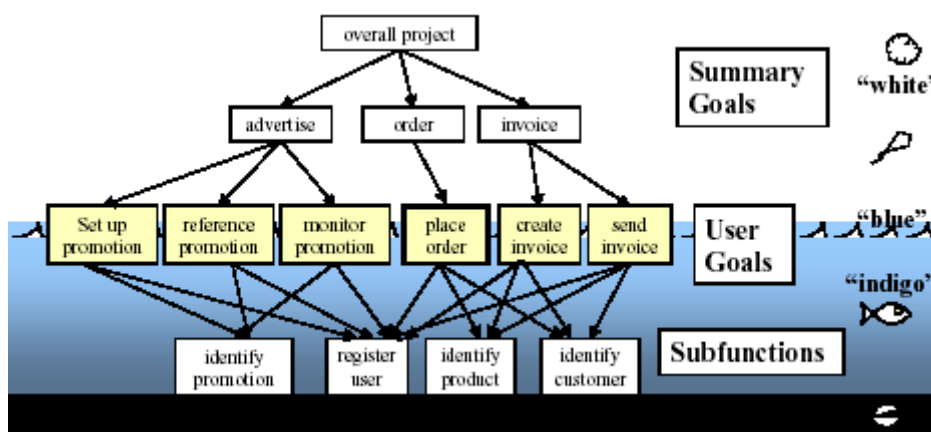


Figura 14. ilustra los nombres y los iconos representantes de los niveles.

3.1 Meta de Usuario (Color Azul, Nivel del Mar 🌊)

La meta de usuario es de gran interés para la captura de requisitos dirigida por casos de uso. La meta de usuario de un actor primario se define como el objetivo para conseguir la finalización de un trabajo. La meta de usuario corresponde a un concepto de ingeniería de proceso de negocio llamado **“proceso elemental de negocio”**.

Una meta de usuario pone atención en las preguntas:

- ¿El actor primario se retira feliz luego de realizar el trabajo?
- ¿La performance del trabajo del usuario depende de cuantas veces haga esto hoy?

También pone atención en la prueba del café de descanso, “luego de finalizar esto, puedo tomarme un tiempo de descanso”. En la mayoría de las situaciones, pasa el siguiente test:

- una persona, un asiento de 2 a 20 minutos

Ni “*realizar el proceso completo de una compra de una subasta on-line*”, ni “*logearse*” generalmente son metas de usuario:

- ✓ Una subasta on-line lleva varios días realizarla, entonces la prueba mencionada anteriormente de una persona en un asiento durante 2 a 20 minutos fallará.
- ✓ Logearse 42 veces en una disputa no hará a las responsabilidades del trabajo de una persona o al propósito de usar el sistema.

Utilizando los mismos métodos de prueba “*Registrar un nuevo cliente*” y “*comprar un libro*”, son candidatos a ser metas de usuario:

- ✓ Registrar 42 nuevos clientes tiene sentido para un vendedor.
- ✓ Comprar un libro puede ser realizado completamente en un asiento entre 2 y 20 minutos.

Por ahora parece ser bastante fácil y no tener dificultad alguna el proceso de reconocer las metas correctas. Pero en la práctica nos encontraremos con un gran número de frases “extrañas” o casos de uso que por alguna razón no lucen correctamente, y esto provocará un gran desconcierto al equipo de trabajo.

Se puede encontrar el camino correcto para identificar metas expresando los niveles de las metas de distintas formas, nosotros utilizaremos una **escala de colores e iconos**.

La escala de colores va desde el color blanco pasando por el azul, índigo, finalizando en el negro. Siendo más claros:

- **Color Blanco** representa los altos niveles:
“*Completar una compra on-line*”
“*Cobrar por un accidente de auto*”
- **Color azul** corresponde a la meta de usuario
“*Registrar nuevos clientes*”
“*Comprar un libro*”
- **Color índigo** representa Bajos niveles
“*Buscar libro*”
- **Color negro**
“*utilizado para indicar que la meta es de tan de bajo nivel que sería un error escribir un caso de uso para esta*”

La idea con la metáfora “**nivel del mar** 🌊” es la que el cielo sigue por kilómetros y kilómetros sobre el “**nivel del mar** 🌊”, y el agua sigue por kilómetros y kilómetros por debajo del “**nivel del mar** 🌊”, pero hay un nivel donde se encuentran el cielo y el mar que es el “**nivel del mar** 🌊”. Haciendo una analogía con el nivel de las metas que persiguen los casos de uso, existen muchos niveles de metas sobre las metas de usuario y muchas debajo de ellas. Por consiguiente, “el nivel del mar” corresponde con las metas de usuario.

Una “**nube** ☁” o un “**barrilete** 🎈” indica que el nivel se encuentra por encima del “**nivel del mar** 🌊” y un “**pez** 🐟” o una “**almeja** 🐚” indica que se encuentra por debajo del nivel del mar.

Es importante destacar que las metas que realmente son más importantes escribir son las metas de usuario y que el sistema queda justificado por el soporte que brinda a las metas de usuario. Una meta de usuario puede ser de la siguiente forma:

“Usted es una telefonista en su puesto de trabajo. Suena el teléfono. Usted atiende. La persona que llamó le dice ‘blablabla...’ usted gira hacia la computadora. Lo que esta en su mente en este momento, es lo que usted necesita resolver ‘X’. Usted trabaja con ambos, con la computadora y con el cliente, por un momento y finalmente resuelve ‘X’. Usted se retira de la computadora, despide a la otra persona y cuelga el teléfono”

X es la meta de usuario, nivel del mar 🌊, o color Azul. Para resolver x, usted resuelve un número de metas de bajo nivel o color índigo. La persona en el teléfono probablemente tenía una meta de alto nivel o color blanco en mente y resolviendo x es solo un paso para lograr la meta.

El nivel del mar 🌊, color Azul o metas de usuario son increíblemente importantes y estas merecen un gran esfuerzo para entenderlas y poder identificar que es lo que las constituyen. Usted podría justificar la existencia del sistema con el soporte a las metas de usuario de los actores primarios. Un corto resumen de una función del sistema es la lista de metas de usuarios que realiza la función. Esta lista es la base para la priorización, servicio, división del equipo, estimación y desarrollo.

Los casos de uso serán escritos con niveles sobre y debajo el nivel del mar por ello es conveniente pensar en el enorme número de metas que existe debajo el nivel del mar y por lo tanto casos de uso “bajo el agua” lo que implica que realmente no tengamos interés en leer ni tampoco escribir casos de uso de bajo nivel debido al nivel de detalle que tendrían los mismos.

3.2 Meta de Resumen (Color blanco, Nube ☁, Barrilete 🎈)

Una meta resumen involucra múltiples metas de usuario. Estas sirven a tres propósitos en la descripción del sistema:

- Muestra el contexto en el cual operan las metas de usuario
- Muestran la secuencia del ciclo de vida de las metas relacionadas
- Proveen una tabla de contenido para ambos niveles de casos de uso, tanto como para casos de uso de nivel mas bajo de color blanco y como para casos de uso azules.

Los casos de uso resumen se encuentran en el color blanco dentro de la escala de colores. Los casos de uso resumen tienen pasos que son blancos, azules u ocasionalmente índigo (ejemplo LogIn).

No hemos encontrado útil distinguir entre distintos niveles de blanco, pero ocasionalmente la persona encargada de las casos de uso piensa “este caso de uso es, definitivamente blanco, ‘está en las nubes’”. En términos de la metáfora “**nivel del mar** 🌊” diremos que la mayoría de los casos de uso son como un “**barrilete** 🎈” justo arriba del “**nivel del mar** 🌊” y el resto están camino hacia las “**nubes** ☁”.

Los casos de uso resumen típicamente se ejecutan durante horas, días, semanas, meses o años.

El ejemplo siguiente muestra un flujo principal de un caso de uso que se ejecuta durante un largo tiempo donde su propósito es agrupar casos de uso azules distribuidos en varios años.

Es necesario que usted pueda reconocer los gráficos como una **casita negra** 🏠 lo que implica que el alcance del caso de uso es la compañía y como una **nube** ☁ que muestra que el nivel del caso de uso es muy alto (se encuentra las nubes). Las frases en *italic* hacen referencia a casos de uso de niveles más bajos.

Caso de Uso: 🏠 “Utilizar una póliza de seguros” ☁

Actor Primario: El cliente

Alcance: La compañía de seguros

Nivel de meta: Resumen (Blanco)

Flujo de sucesos principal

1. Cliente obtiene un precio por una póliza
2. Cliente compra póliza
3. Cliente hace una demanda contra la póliza
4. Cliente cierra póliza

3.2.1 Casos de uso de gran altura


Recomendamos escribir los casos de usos mas altos para cualquier sistema que usted diseñe. Daremos una descripción mas precisa para buscar estos casos de uso:

1. Comenzar por la meta de usuario

2. Preguntarse ¿ A qué actor primario (preferentemente fuera de la organización) realmente sirve esta meta?
AA es el “actor primario ulterior” de los casos de uso de los cuales tratamos de reunir.
3. Encontrar el alcance de diseño mas alto “S” tal que el AA todavía quede afuera de “S”. Nombre el alcance “S”. Nosotros encontramos tres de alcance de diseño más altos alcance:
 - La compañía
 - La combinación de los sistemas de software
 - El sistema de software que está siendo diseñado
4. Encontrar las metas de usuario que tienen como actor primario ulterior a AA y alcance de diseño S.
5. Trabaje en las metas resumen GG que el Actor AA tiene contra el sistema S. Este caso de uso agrupa varios casos de usos de nivel del mar.

En todo lo dicho, hay usualmente solo 4 o 5 de estos casos (GG) de gran altura, aún en los sistemas más grandes. Los casos de uso de gran altura resumen los intereses de 3 o 4 actores primarios ulteriores AA:

- El cliente, como el actor primario mas alejado de la compañía
- El departamento de marketing, como el actor primario mas alejado de la combinación de los sistemas de software.
- El departamento de seguridad, como el actor primario mas alejado del sistema de software mismo.

Estos casos de uso de gran altitud son muy útiles en la integración del trabajo y es altamente recomendable escribirlos, por las razones antes mencionadas. Estos no son proveídos por el equipo con los requerimientos funcionales del sistema a ser desarrollado ya que residen en las metas de usuarios (nivel del mar ).


3.3 Subfunciones (Color índigo, bajo el agua , Almejas)

Los niveles de sub-función son aquellos requeridos para llevar a cabo las metas de usuario. Incluya estos niveles solo cuando se vea obligado a hacerlo, por ejemplo en las ocasiones que éstos son necesarios para mejorar la lectura, o porque muchas otras metas los utilizan.

Ejemplos de casos de usos de sub-funciones pueden ser:

- “Caso de uso: encontrar un producto”
- “Caso de uso: encontrar un cliente”
- “Caso de uso: guardar como.....(un archivo)”

Casos de uso de sub-funciones se encuentran bajo el agua, en el sector de color índigo en la escala de colores.

Existen algunos casos de uso de sub-funciones que realmente se encuentran “bajo el agua”, es decir en las profundidades del mar y se sitúan en el barro dentro de la escala. El color de estos últimos es negro, para significar que **“esto son de tan bajo nivel, que expandirlos sería un error, ya que complicaría la interpretación”**. Haciendo una analogía con los objetos que se encuentran enterrados en el fondo del mar llamaremos a éstos almejas . Es conveniente nombrar de alguna forma especial a estos casos de uso de ultra-bajo-nivel, para que cuando alguien escriba uno, usted podrá indicar que este tipo de casos de uso no debería ser escrito.

Los casos de uso azules tienen pasos índigo, y los pasos índigo tienen pasos mas profundos color índigo como lo muestra la figura 15.

En ésta figura también encontrará el nivel mas alto de sus metas contestando su pregunta ¿porque el actor hace esto?. Esta técnica la veremos mas adelante en el punto 2.5.1. Encontrar la meta de usuario.

3.4 Conclusión acerca de los niveles de metas

Por ahora los puntos más importantes acerca de los niveles de metas son:

- Poner mucha energía en la detección acerca de los niveles de metas del nivel del mar. Estos niveles son realmente los importantes.
- Escribir los pocos casos de usos de gran altura para proveer el contexto para los demás casos de usos.
- No le dé demasiada importancia si su frase mas frecuente en las oraciones de los requerimientos del sistema “Hacer.....” aparece o no como un titulo de un caso de uso.

3.5 Encontrando el nivel correcto de las metas

Lo más complicado de realizar la captura de requisitos de software mediante casos de uso es encontrar el nivel correcto de las metas de los mismos. Haciendo hincapié en esto ultimo plantearemos los siguientes puntos:

- Encontrar la meta de usuario
- Use de 3 a 10 pasos por caso de uso

3.5.1 Encontrar la meta de usuario

En todos los niveles de las metas, solo un nivel se destaca en carácter de importancia por encima de los otros.

Usted está describiendo un sistema, tanto de negocio o como de computadora, la persona que a usted le merece importancia es quien usa el sistema. Esta persona quiere algo de su sistema en el instante. Luego de obtenerlo, esta persona puede continuar y realizar algo más. ¿Que es lo que buscan de su sistema, en ese mismo instante?

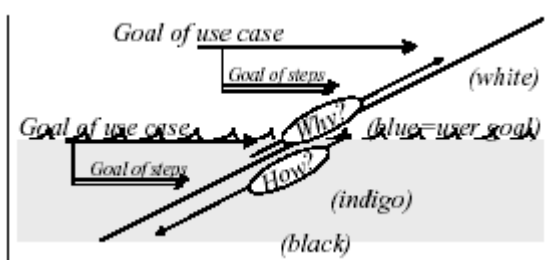
Este nivel tiene muchos nombres. Desde el punto de vista del modelado de proceso de negocio, se los llama **“Proceso elemental de negocio”**, visto desde el punto de vista de casos de uso se los llama **“metas de usuario”**.

La primera pregunta es ¿ Es esto lo que realmente quiere el actor primario de nuestro sistema, en ese mismo instante? Para muchos la primera respuesta es NO. Muchos principiantes describen en sus primeros bosquejos casos de uso bajo nivel o bajo el agua, pensando que están en el nivel del mar o meta de usuario. Para encontrar un nivel mas alto de la meta es decir la meta de usuario, debe preguntarse:

- ¿Qué busca realmente el actor primario?
- ¿Por qué esta haciendo esto el actor?

La respuesta, debería ser la meta real de del actor. Pero en caso de no encontrarla debe preguntarse la pregunta de nuevo, hasta que encuentre la meta real del usuario. Lo interesante de esta prueba es que aun siendo subjetiva, distintas personas pronto convergen en resultados muy similares.

Figura 15. Pregunte ¿ por que? Para cambiar de nivel



3.5.2 Use de 3 a 10 pasos por caso de uso

Este punto se centrará en la longitud del caso de uso.

Muchos casos de uso bien escritos tienen de 3 a 8 pasos. Nunca hemos visto un caso de uso mas largo que 11 pasos que no haya quedado mejor cuando fue acortado. No hay fundamentos teóricos del porque de esto, pero es posible que sea que las personas no toleran o no pueden pensar en término de procesos que tomen mas de 10 pasos.

Utilice lo anterior para mejorar su escritura, si usted tiene mas de diez pasos, probablemente ha incluido detalles de la interface de usuario, o ha escrito pasos de acciones a muy bajo nivel. Para corregir su escritura:

- Remueva los detalles de interfaces de usuario
- Eleve el nivel de la meta con la pregunta ¿Por qué? Para encontrar el nivel próximo mas alto
- Combine pasos.

4. Precondiciones, Disparadores, Garantías

4.1 Precondiciones

La precondición de un caso de uso anuncia lo que el sistema asegurará que se ha cumplido antes de permitir el comienzo del caso de uso. Luego de asegurarse que la precondición es verdadera, esta no será revisada durante el caso de uso.

Un ejemplo común es:

“Precondición: el usuario ya se ha logeado y el logeo ha sido válido”.

Generalmente tener una precondición en un caso de uso B, indica que algún otro caso de uso A se encuentra corriendo y necesita la ejecución de B, por lo que B establece las condiciones que se deben cumplir para su ejecución.

Se puede decir que el **Caso de Uso 3, “realizar un pedido”**, depende de una precondición (“loguearse”). Inmediatamente miraría para ver que Caso de Uso contiene esta funcionalidad (quizás el **Caso de Uso 2** es llamado logearse). Un buen método es crear un caso de uso de un nivel más alto que contenga ambos casos de uso 2 y 3, de esta manera el lector del caso de uso puede ver la manera en que se enlazan. En el ejemplo antes mencionado, el caso de uso resumen podría ser el caso de uso 1, “Usar la aplicación”. Quedando de su estructura de la siguiente manera:

Caso de Uso 1: Usar la aplicación

Nivel: Resumen 

Precondición: No hay

1. Empleada logearse.
2. Empleada realiza un pedido.

Caso de Uso 2: Logearse

Nivel: Subfunción 

Precondición: No hay

...

Caso de Uso 3: Realizar un pedido

Nivel: Usuario 


Precondición: Empleada se encuentra logeada

...

NOTA: No todo el mundo trabaja de esta forma escribiendo el caso de uso de mas alto nivel para mostrar como se juntan los niveles más bajos. Es decir usted podría en el caso del ejemplo anterior adoptar su propia forma de trabajo solamente definiendo los casos de uso 1 y 2, pero debería estar capacitado para deducir a partir de las descripciones en forma narrada si la precondición es correcta y asegurar que será cumplida.

Continuemos con el ejemplo para mostrar la forma en que un caso de uso pone una condición a un paso, y la condición depende de otros pasos. Una vez más, el sub-caso de uso esperará que la condición se cumpla y no chequeará por errores durante su ejecución:

Caso de Uso 3: Realizar un pedido


Nivel: Usuario 

Precondición: El Empleado se encuentra logeado

1. El empleado identifica al cliente, el sistema se posiciona en el registro del cliente.

2. El empleado entra la información del pedido.
3. El sistema calcula el precio.
- ...

Caso de uso 4: Calcular precios

Nivel: Subfunción 

Precondición: El cliente es admitido y conocido para el sistema, el contenido del pedido es conocido.

1. El sistema calcula el precio base para el pedido.
2. El sistema calcula el descuento para el pedido.
- ...

Este ejemplo ilustra como un caso de uso se basa en la información del un caso de uso invocado.

El escritor de calcula precio declaró la información que está disponible y puede seguir adelante y hacer referencia a la información del cliente.

Nota: en el ejemplo utilizado el lector no estará de acuerdo acerca del caso de uso calcular precio. Hemos declarado este caso de uso como de color índigo, pero por ahora el lo que respecta a la escritura hay poco que lo justifique aún teniendo su propio caso de uso. Si nosotros, como escritores, no describimos las interacciones complicadas con el usuario, o casos de falla interesantes, lo reclasificaríamos cambiándolo de índigo a negro (pescado a almeja). Esta es la señal para combinar el texto de calcular precio dentro del caso de uso realizar un pedido y eliminar el caso de uso calcular precio.

Escribir una precondición es escribir como afirmaciones simples acerca del estado del mundo en el momento en que el caso de uso es abierto. Los ejemplos apropiados son:

"Precondición: El usuario está logeado."

"Precondición: El cliente ha sido validado."

"Precondición: El sistema ya ha localizado, la póliza del cliente."

Un error común es escribir dentro de la precondición algo que frecuentemente se cumple, pero no es necesario que se cumpla.

Suponga que estamos escribiendo la Petición de un resumen de cuenta, donde el actor primario es el solicitante. Seguramnete pensaríamos que el solicitante estaría seguro de haber hecho al menos un pedido o tener una factura antes de preguntar por el resumen de cuenta. Sin embargo, este no siempre es el caso. El sistema no puede asegurarlo, y de hecho esto no es esencial. El solicitante debe estar habilitado para preguntar por su resumen de cuenta en cualquier momento. Entonces esta mal escribir "Precondición: El solicitante ha realizado una factura."

4.2 Garantías Mínimas

Las mínimas garantías son las promesas mínimas que el sistema realiza a los interesados en el sistema, particularmente para cuando la meta del actor primario no es cumplida. Ellas se mantienen cuando la meta es cumplida, por supuesto, pero ellas se transforman en intereses reales cuando la meta principal es abandonada. Muchas de las veces, dos o más personas tienen que ser conducidos hacia las garantías mínimas.

No se moleste listando en la sección de mínimas garantías todas las maneras que un caso de uso puede fallar. Hay docenas de maneras como fallar, y estas tienen pocas cosas en común. Todo lo referente a las condiciones de fallas y su manejo se muestra en la sección de extensiones, y esto es cansador y propenso a error tratar de mantener dos lista sincronizadas. El propósito de esta sección de la plantilla es anunciar lo que promete el sistema.

La garantía mínima más común es "que alcance tiene el log de transacciones del sistema" no piense que una falla de una transacción de log es obvio o trivial. Los logs de transacciones de los sistemas son frecuentemente olvidados en la descripción de requerimientos, y a veces redescubiertos por los programadores. El logeo es crucial para los dueños del sistema como para el usuario. Las personas interesadas en el sistema los

usan para resolver discusiones. El escritor de casos de uso deberá descubrir la necesidad de logs de transacciones para investigar los intereses de algunas personas o cuando hay una demasiadas condiciones de falla.

La garantía mínima es escrita como un número de simples afirmaciones que serán cumplidas en el final de cualquier ejecución del caso de uso. Estas muestran los intereses de cada persona que queda satisfecha.

Garantía Mínima: El pedido comenzará solo si el pago se ha recibido.

Garantía Mínima: Si la información mínima no fue capturada, la demanda parcial ha sido descartada y no se dejan rastros de log de la llamada. Si la mínima información fue capturada, entonces la demanda parcial ha sido guardada y se hacen los registros.

Nota: La prueba de éxito/falla para la mínima garantía es que los interesados en el sistema acordarán que sus intereses han sido protegidos aún bajo condiciones de falla de su meta.

4.3 Garantía de Éxito

La garantía de éxito manifiesta los intereses de las personas que quedarán satisfechas después de una finalización exitosa del caso de uso, ya sea por el final del escenario principal de sucesos o por el final de la sección de extensines. La garantía de éxito generalmente es escrita para agregar algunas condiciones extras a la Garantía Mínima, tales condiciones adicionales incluyen al menos el estado de la meta del título de caso de uso.

Como la garantía mínima, la garantía de éxito es escrita como simples afirmaciones que se cumplen al final de una ejecución de un caso de uso. Estas muestran los intereses de las distintas personas que son satisfechas.

Ejemplos :

Garantía de Exito: Al denunciante se le pagara la cifra acordada, la demanda se cerró, el registro de la transacción fue asentado.

Garantía de Exito: El archivo será guardado.

Garantía de Exito: El sistema iniciará un pedido del cliente, se recibirá la información del pago, y se registrará la solicitud del pedido.

Nota: La prueba éxito/falla para la garantía de éxito es que las personas acordarán, desde la lectura de la sección garantía de éxito, que sus intereses han sido satisfechos.

La mejor forma de descubrir la garantía de éxito es preguntar ¿Qué haría a esta persona infeliz al final de la ejecución exitosa? Esta pregunta es fácil de responder, entonces para la garantía de éxito escriba lo negativo de esta respuesta.

4.4 Disparadores

El disparador manifiesta el evento que hace que el caso de uso comience. A veces el disparador precede el primer paso de caso de uso. A veces el disparador es el primer paso del caso de uso. Hasta ahora, no hemos visto una regla convincente que se aplique en todos los casos. Tampoco hemos tenido alguna confusión eligiendo una manera u otra.

Aui tenemos algunos ejemplos:

Considere una persona usando un cajero automático. El cajero despierta de su inactividad solo cuando la persona inserta su tarjeta. Esto no es insignificante para decir que el disparador es cuando alguien decide usar el cajero. El disparador “Cliente inserta tarjeta”, es también el primer paso en el caso de uso.

CASO DE USO: Usar el Cajero Automático

Disparador: Cliente inserta la tarjeta

1. Cliente inserta la tarjeta con identificación bancaria, (cuenta bancaria y pin encriptado)
2. El sistema valida...

Considere, sin embargo, un oficinista todo el día en una estación de trabajo que muestra distintos iconos para ejecutar diferentes aplicaciones. El disparador es que llama un cliente con una solicitud en particular.

Esto puede ser escrito utilizando la ambas formas, pero utilizaremos la segunda forma.

CASO DE USO: Registrar una queja

Disparador: Cliente llama para realizar en una queja.

1. El oficinista llama a la aplicación.
2. El sistema trae la lista de quejas recientes

...

5- Escenarios y Pasos

Un conjunto de casos de uso es una historia de actores primarios persiguiendo metas. Cada caso de uso individual tiene un relato que muestra al sistema llegando a la meta o abandonándola. El relato es presentado como un escenario principal y un conjunto de fragmentos de escenarios como extensión del primero. Cada escenario o fragmento comienza con una condición disparadora y continúa hasta que muestra la concreción o el abandono de la meta. Las metas son de diferentes niveles, como hemos visto, pero nosotros usamos la misma forma de escritura para describir cualquier nivel de meta en cualquier nivel de escenario.

5.1 El escenario principal, escenarios

5.1.1 El escenario principal como el caso simple

Usualmente explicamos las cosas a otras personas empezando con una descripción fácil de entender y entonces ampliamos, “ Bueno, actualmente hay una pequeña complicación. Cuando esto y esto ocurre, lo que pasa es que....”. Las personas hacen muy bien este tipo de explicaciones y es la forma en que escribimos el relato de un caso de uso. Primero escribimos una descripción de principio a fin de un simple y típico escenario en el cual la meta del actor primario es conseguida y los intereses de las personas que son satisfechos. Este es el *escenario principal*.

Todas las otras formas de alcanzar la meta, y el manejo de las fallos, son descriptos en las extensiones del escenario principal.

5.1.2 Estructura común

El escenario principal y las extensiones tienen la misma estructura. La misma consiste en:

- Una condición bajo la cual el escenario se ejecuta. Para el escenario principal es la precondition más el disparador. Para un escenario extendido, es la condición de extensión (quizás con el número de paso o el lugar del escenario donde se aplica la condición).
- Una meta a alcanzar. Para el escenario principal, es exactamente el nombre del caso de uso, satisfaciendo, por supuesto, los intereses de las personas. La meta del escenario extendido es también completar la meta del caso de uso o volver a unirse al escenario principal después de manejar la condición.
- Un conjunto de pasos con acciones. Éstos forman el cuerpo del escenario, y siguen las mismas reglas en cada escenario o fragmento de escenario.
- Una condición de fin. La meta es alcanzada al final del escenario principal. Un fragmento de escenario puede terminar con la meta alcanzada o abandonada.
- Un posible conjunto de extensiones, escritas como fragmentos de escenarios. Las extensiones al escenario principal van en la sección Extensiones de la plantilla del caso de uso. Las extensiones de extensiones van directamente en línea, dentro o justo después de las extensiones.

Aquí hay un ejemplo para mostrar la similitud de sus estructuras:

CU: Comprar mercaderías por la web

Precondición: El usuario tiene abierto el PAF (Consejero Personal de Finanzas)

Disparador: El usuario selecciona “comprar mercaderías”

Escenario Principal

- 1- El usuario selecciona comprar mercaderías por la web
- 2- PAF obtiene el nombre del usuario del sitio a usar (Amazon.com por ejemplo).
- 3- PAF establece la conexión al sitio, reteniendo el control
- 4- El usuario busca y compra mercaderías del sitio.
- 5- PAF intercepta las respuestas del sitio, y actualiza el portafolio del usuario.
- 6- PAF muestra el nuevo estado del portafolio del usuario.

Extensiones:

3a- Falla la conexión a la web:

3a1- El sistema informa la falla al usuario, volviendo al paso previo.

3a2- El usuario abandona el caso de uso, o trata de nuevo

En este capítulo veremos en detalle el cuerpo del escenario, el cual consiste en pasos con acciones.

5.1.3 El cuerpo del escenario

Cada escenario o fragmento es escrito como una secuencia de acciones para alcanzar las metas de los actores. Decimos *secuencia* por conveniencia, pero podemos agregar notas para describir que esos pasos pueden ir en paralelo, tomados en diferente orden, repetidos, y hasta algunos son opcionales. Cada paso describe:

- una interacción entre dos actores (“El cliente ingresa domicilio”).
- una validación para proteger un interés de una persona (“El sistema valida el password”),
- un cambio interno para satisfacer un interés de una persona (“El sistema deduce importe de la cuenta”)

Aquí hay un ejemplo de un típico Escenario Principal. Notará que los pasos 1,3,5-8 son interacciones, el paso 4 es una validación, y los pasos 2 y 9 son cambios internos.

- 1- El empleado ingresa el número de la póliza o bien el nombre y fecha de incidente.
- 2- El sistema obtiene la información disponible de la póliza e indica si el reclamo corresponde a la póliza.
- 3- El empleado ingresa la información básica del siniestro.
- 4- El sistema confirma que no haya reclamos contrarios y asigna un número de reclamos
- 5- El empleado continúa ingresando la información del siniestro específica para el reclamo.
- 6- El empleado obtiene más información de la cobertura de otros sistemas.
- 7- El empleado selecciona y asigna un mediador.
- 8- El empleado confirma que ha finalizado
- 9- El sistema salva y el acuse de recibo es enviado al agente.

Cada paso es escrito para mostrar una acción simple. El mismo estilo de escritura es usado para los pasos en cualquier parte de cualquier caso de uso, ya sea escenario principal o extensión, caso de uso del negocio o del sistema, de alto o bajo nivel.

5.2 Pasos

Los pasos que hacen a un caso de uso bien descripto, están escritos en forma gramatical, una acción simple en la cual un actor ejecuta una tarea o pasa información a otro actor:

“El usuario ingresa nombre y apellido”

“En cualquier momento, el usuario puede pedir que le devuelvan el dinero”

“El sistema verifica que el nombre y la cuenta están actualizados”

“El sistema actualiza la cuenta del cliente para reflejar el cambio”

La mayoría de las veces, el tiempo puede ser omitido, ya que los pasos generalmente se suceden unos tras otro. Hay muchas pequeñas variantes en la forma de escribir los pasos, sin embargo siempre escriba preservando las siguientes características en cada paso.

5.2.1 Directrices para los pasos

Directriz 1: Usar gramática simple.

La estructura de la oración debe ser simple:

“Sujeto...verbo...objeto directo...frase preposicional.”

Ejemplo:

“El sistema...deduce...el importe...de la cuenta.”

Mencionamos esto porque muchas personas olvidan accidentalmente el primer sustantivo. De esa forma, no está claro quien está controlando la acción. Si sus oraciones están mal estructuradas, la historia se hace difícil de seguir.

Directriz 2: Muestra claramente “quien tiene el balón”

Una imagen visual útil es un partido de fútbol. La persona 1 se la pasa a persona 2, esta la retiene un momento, entonces la pasa a la persona 3. Ocasionalmente se vuelve trabado y uno de los jugadores limpia la jugada.

Un escenario tiene la misma estructura. En cada paso un actor “tiene el balón”. Ese actor será el sustantivo de la oración, el primer actor mencionado. El “balón” es el mensaje y los datos que son pasados de actor a actor. Algunas veces, el paso termina con otro actor teniendo el balón. Pregúntese “¿Al final de la oración, quien tiene el balón?”. Debe ser siempre claro en la escritura “quien tiene el balón”.

Directriz 3: Está escrito con un punto de vista de ojos de pájaro.

Los escritores de casos de uso principiantes, particularmente programadores que han sido asignados a escribir casos de uso, usualmente escriben el escenario como lo ve el sistema, mirando el mundo exterior e interactuando con él. Las oraciones tienen la apariencia “Obtener tarjeta y numero de clave. Deducir importe de la cuenta”.

Lo correcto es escribir el caso de uso con ojos de pájaro:

“El cliente ingresa tarjeta y clave”

“El sistema deduce el importe de la cuenta”

Directriz 4: Muestra claramente el avance del proceso

El progreso hecho en cada paso es relativo a cuán alta o baja es la meta del caso de uso. En un caso de uso resumen, el paso probablemente se mueve una meta de usuario entera. En un caso de uso subfunción se mueve hacia delante mucho menos. Si nosotros vemos el paso “El usuario presiona la tecla tab”, o bien estamos viendo un caso de uso en índigo profundo, o el escritor simplemente eligió una muy pequeña acción para describir.

El error de elegir pasos muy pequeños se descubre en la longitud del caso de uso. Si un caso de uso tiene de 13 a 17 pasos, es bastante probable que las oraciones no avancen lo suficiente hacia la meta. El caso de uso es más fácil de leer, más claro, y contiene la misma información esencial cuando combinamos esos pequeños pasos en uno que avance más directamente hacia la meta. Raramente se encuentra un caso de uso bien escrito con más de 9 pasos en el escenario principal.

Para encontrar la meta de mayor nivel de un paso, pregúntese “¿Por qué el actor está haciendo esto?”. La respuesta a esa pregunta es probablemente la meta que necesita para ese paso, aunque deberá preguntárselo varias veces para obtener la respuesta que usted quiere. Aquí hay un ejemplo:

“El usuario presiona la tecla tab”

¿Por qué el usuario está presionando la tecla? Para alcanzar el campo domicilio. ¿Por qué está tratando de alcanzar el campo? Porque tiene que ingresar su nombre y su domicilio antes que el sistema haga algo. Ah! Entonces ella quiere que el sistema haga algo (probablemente el caso de uso mismo), y para ello tiene que ingresar su nombre y domicilio. Entonces, la oración que avanza en la ejecución del caso de uso hacia el cumplimiento de la meta es:

“El usuario ingresa su nombre y domicilio”

Directriz 5: Muestra el intento del actor, no el movimiento.

Describir los movimientos del usuario operando la interfaz del sistema es uno de los más comunes y severos errores, y está ligado a escribir metas en un nivel muy bajo. Llamamos a estas descripciones como “descripciones de diálogo”. Las mismas hacen la documentación de requerimientos deficiente de tres formas: más larga, frágil y con demasiadas restricciones.

- Documentos más largos son más difíciles de leer y más costosos de mantener.
- El diálogo descripto no es probablemente un requerimiento, es probablemente sólo la forma en que el escritor imagina la interfaz de usuario en ese momento.

- El diálogo es frágil, en el sentido que pequeños cambios en el diseño del sistema invalidarán la escritura.

Es trabajo del diseñador de interfaz de usuario el inventar una que sea efectiva y permita al usuario llevar a cabo el intento que el caso de uso describe. La descripción de movimientos particulares pertenece a esa tarea de diseño, no a la documentación de requerimientos funcionales.

En la documentación de requerimientos, estamos interesados en la descripción *semántica* de la interfaz, que indique el intento del usuario, y en dar sólo un resumen de la información que es pasada de un actor a otro.

Se utiliza el término *casos de uso esenciales* para designar aquellos en los que estamos interesados: casos de uso del sistema con nivel del mar escritos con descripciones semánticas de la interfaz. Típicamente, todos los datos pasados en una dirección son recogidos en sólo un paso. Aquí damos un ejemplo de un caso de uso mal escrito arreglado:

Antes:

1. El sistema solicita el nombre
2. El usuario ingresa el nombre
3. El sistema solicita el domicilio
4. El usuario ingresa el domicilio
5. El usuario presiona OK
6. El sistema presenta el perfil del usuario

Después:

1. El usuario ingresa nombre y domicilio
2. El sistema presenta el perfil del usuario

Directriz 6: Contiene un conjunto razonable de acciones.

Ivar Jacobson ha descrito un paso en un caso de uso como representando una transacción. Con esta frase él captura cuatro partes de una interacción combinada (Figura 16):

1. El actor primario envía una petición y datos al sistema.
2. El sistema valida la petición y los datos.
3. El sistema altera su estado interno.
4. El sistema responde al actor con el resultado.

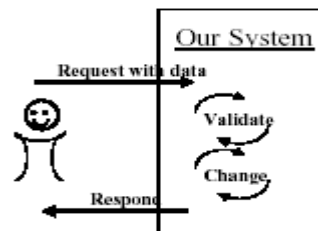


Figura 16

Usted puede escribir cada parte como un paso separado, o combinarlos de varias maneras, o hasta poner los cuatro en un solo paso. Lo mejor depende de cuán complicada sea cada parte y dónde ocurre el corte natural en el proceso.

Como ejemplo, aquí hay cinco variantes para considerar. Ninguna es incorrecta, sin embargo consideramos la versión 1 demasiado complicada para leer. Nos gusta la versión 2 donde las partes son simples, pero las encuentro demasiado largas para trabajar en esta instancia. La versión 3 es nuestra preferida para este ejemplo. La versión 4 es también buena, pero encontramos los pasos un poco pequeños, haciendo el escenario demasiado extenso para nuestro gusto. La versión 5 tiene la ventaja de tener pasos que son unidades de testeo separadas, posiblemente para satisfacer una más formal situación de desarrollo.

Versión 1:

1. El cliente ingresa el número de orden. El sistema detecta que es el número ganador del mes, registra el usuario y el número de orden como ganadores del mes, envía un correo electrónico al gerente de ventas, felicita al cliente y le da las instrucciones sobre cómo obtener el premio.

Versión 2:

1. El cliente ingresa el número de orden.
2. El sistema detecta que es el número ganador del mes, registra el usuario y el número de orden como ganadores del mes, envía un correo electrónico al gerente de ventas, felicita al cliente y le da las instrucciones sobre cómo obtener el premio.

Versión 3:

1. El cliente ingresa el número de orden.
2. El sistema detecta que es el número ganador del mes.
3. El sistema registra el usuario y el número de orden como ganadores del mes, envía un correo electrónico al gerente de ventas, felicita al cliente y le da las instrucciones sobre cómo obtener el premio.

Versión 4:

1. El cliente ingresa el número de orden.
2. El sistema detecta que es el número ganador del mes.
3. El sistema registra el usuario y el número de orden como ganadores del mes, envía un correo electrónico al gerente de ventas.
4. El sistema felicita al cliente y le da las instrucciones sobre cómo obtener el premio.

Versión 5:

1. El cliente ingresa el número de orden.
2. El sistema detecta que es el número ganador del mes.
3. El sistema registra el usuario y el número de orden como ganadores del mes.
4. El sistema envía un correo electrónico al gerente de ventas.
5. El sistema felicita al cliente y le da las instrucciones sobre cómo obtener el premio.

Directriz 7: No es “controla si”, es “valida”

Una de las tres clases de acciones es cuando el sistema verifica que se cumpla alguna regla del negocio. Usualmente las personas escriben que el sistema *controla* la condición. Este no es un buen verbo para representar la acción. No avanza el proceso hacia el cumplimiento de la meta, y deja abierto el resultado del control. Usted inmediatamente tiene que escribir “Si pasa el control” y “Si falla el control....”.

Vamos a usar la técnica de preguntar por qué para encontrar una frase mejor. ¿Por qué el sistema está controlando esta condición? Está validando o garantizando algo. Esos son buenos verbos para usar. Entonces reemplace “El sistema controla si el password es correcto” por “El sistema valida que el password es correcto”.

Deje que la presencia de la palabra “si” le lleve a usted a recordar esta directriz. Cada vez que vea “Si (la condición).. entonces..”, mire la oración anterior. Es muy probable que diga *controla*. Reemplace la primera oración con *valida*, y haga la segunda oración una acción simple sin la palabra “si”. Aquí hay un ejemplo:

Antes:

2. El sistema controla si el password es correcto.
3. Si es así, el sistema presenta las acciones disponibles al usuario.

Después:

2. El sistema valida que el password es correcto.
3. El sistema presenta las acciones disponibles al usuario.

Note que la descripción del segundo caso describe el escenario sucediendo. También lleva al lector a preguntar “¿Pero, si el password no es válido?”. El lector mirará la sección de extensiones buscando la que diga “El password no es válido”. Esto le da al caso de uso un ritmo consistente que hace al mismo fácil de leer y revisar.

Directriz 8: Es opcional mencionar el tiempo

La mayoría de los pasos vienen directamente del paso previo. Ocasionalmente necesitará decir algo como: “En algún momento entre el paso 3 y 5, el usuario...”

“Tan pronto como el usuario haya..., el sistema...”

Siéntase libre de poner el tiempo, pero sólo cuando sea necesario. Usualmente el paso del tiempo es obvio, y no es necesario mencionarlo.

Directriz 9: Frase “El usuario hace que el sistema A ...del sistema B”

Esta es la situación que se puede presentar. Usted quiere que el sistema bajo diseño traiga información del sistema B, o que tenga una interacción con el sistema B, Esto sólo debería hacerse cuando el actor primario

indica cuando hacerlo. No podemos escribir “El usuario presiona el botón Recuperar, en ese momento el sistema recupera los datos del sistema B”. Estaríamos describiendo la interfaz del usuario.

Podemos usar, en cambio, dos pasos:

1.El usuario indica al sistema recuperar los datos del sistema B

2.El sistema recupera los datos del sistema B

A pesar de ser aceptable, es redundante. Es mejor escribir:

1.El usuario hace que el sistema recupere los datos del sistema B

Con este pequeño cambio en la escritura, indicamos que el usuario tiene el control, que el balón es pasado del usuario al sistema A, y este al sistema B, mostrando las responsabilidades de los tres sistemas. Los detalles de cómo el usuario indica la acción no se especifica.

Directriz 10: Frase “Hacer los pasos x-y hasta condición”

En ocasiones, desearemos marcar que algunos pasos pueden repetirse. En este caso debemos escribirlo en prosa simple, no usando formalismos de programación. Solamente escriba que el o los pasos serán repetidos. Si sólo hay un paso a ser repetido, puede poner la repetición dentro del paso:

“El usuario selecciona uno o más productos”

“El usuario busca entre varios catálogos de productos hasta encontrar el que quiere usar”

Si varios pasos son repetidos, puede escribir la repetición antes o después de los pasos repetidos. Nosotros escribimos la repetición después de los pasos, para hacer el escenario un poco más fácil de leer. Sin embargo, la otra forma también funcionará.

1. El cliente completa su identificador de cuenta, o bien su nombre y domicilio.

2. El sistema presenta la información de las preferencias del usuario.

3. El usuario selecciona un ítem a comprar, marcándolo para adquirirlo.

4. El sistema agrega el ítem al “carrito de compras” del usuario.

El cliente repite los pasos 3-4 hasta que indica que ha terminado.

5. El cliente compra los ítems en el “carrito de compras”.

Note que no es necesario numerar la sentencia de la repetición, ni tampoco una expresión que abra la repetición. Ambas cosas harían confusa la escritura, haciendo el escenario más difícil de leer.

Una variante de “Hacer los pasos x-y hasta condición” es “Los pasos x-y pueden ocurrir en cualquier orden”. Esto funciona bien poniéndolo antes de los pasos afectados.

1. El cliente se logea

2. El sistema presenta los productos y servicios disponibles.

Los pasos 3-5 pueden aparecer en cualquier orden.

3. El usuario selecciona los productos a comprar.

4. El usuario especifica la forma de pago.

5. El usuario da la dirección de entrega.

6. El usuario indica que la sesión de compra está completa.

7. El sistema inicia la orden, con los productos seleccionados a ser cargados contra la forma de pago y a ser entregado en la dirección de entrega.

6. Extensiones

Un escenario es una secuencia de pasos. Sabemos que un caso de uso debe contener todos los escenarios, los exitosos y los que fallan. Ya sabemos cómo escribir el escenario principal. Ahora necesitamos una forma de agregar todos los otros escenarios.

Podríamos escribir cada escenario individualmente, sin embargo es muy difícil de mantener, ya que cada cambio debería ser copiado a los otros escenarios que contienen ese texto.

Una alternativa es escribir sentencias *si* a lo largo del texto: “*Si el password es correcto, el sistema..., en caso contrario el sistema...*”. Esto es perfectamente válido, y algunas personas escriben casos de uso de esa manera. Sin embargo, a los lectores les cuesta las condiciones *si*, especialmente cuando hay un *si* dentro de otro *si*. Se pierde rastro del comportamiento después de dos bifurcaciones *si*, y la mayoría de los casos de uso contienen muchos puntos de bifurcación.

Otra alternativa es escribir el escenario principal como una secuencia simple corriendo desde su inicio hasta su terminación, y entonces escribir una *extensión* al escenario por cada punto de bifurcación. Esta es, a nuestro parecer, la mejor elección de las tres.

Las extensiones trabajan de esta forma. Debajo del escenario principal, por cada lugar donde el comportamiento puede bifurcar por una condición particular, se escribe la condición y los pasos que la manejan. Muchas extensiones terminan con una simple reinserción en el escenario principal.

La extensión es como un caso de uso. Contiene una secuencia de pasos describiendo qué pasa bajo esas condiciones. Termina cuando se alcanza o abandona la meta de la extensión. Puede haber extensiones de extensiones para manejar los *si* encontrados durante el camino. En estas extensiones residen los más interesantes requerimientos del sistema. El escenario principal es usualmente conocido por el equipo de desarrollo. El manejo de fallos generalmente usa reglas del negocio que los desarrolladores desconocen. Los escritores de requerimientos frecuentemente tienen que hacer alguna investigación para determinar la respuesta correcta del sistema. Usualmente, esa investigación introduce un nuevo actor, nuevo caso de uso, o una nueva condición de extensión. Aquí tenemos una típica discusión para ilustrar:

“Supongase que hay una repentina falla de la red. ¿Qué debemos hacer?”

“El sistema debe llevar un log”

“OK, pero si la red cayó, ¿qué se supone que sucede cuando se levanta?”

“Creo que tendríamos que agregar un nuevo caso de uso para *El sistema se reinicia después de una falla de red*. El sistema se reiniciará, obtendrá el log, y completará o abortará las transacciones”.

“Sí, pero ¿que pasa si el log esta defectuoso?. ¿Qué se supone que haga el sistema?”

“No sé. Lo dejemos como cuestión abierta y continuamos”

En esta conversación, se descubrió la necesidad de un nuevo caso de uso y la necesidad de investigar una política del negocio. No piense que estas son extensiones innecesarias para discutir. Algunos programadores se encuentran con estas condiciones, mientras programan. Este es un momento costoso para descubrir nuevas reglas del negocio que se deban investigar. Durante la etapa de requerimiento es el mejor momento para descubrirlas. Recomendamos trabajar en tres fases:

- Piense e incluya cada posibilidad que usted y sus colegas conjeturen.
- Evalúe, elimine, y combine las ideas de acuerdo a las directrices que se describen más abajo.
- Entonces trabaje en cómo el sistema debería manejar cada una de esas condiciones.

6.1 Condiciones de extensión

Las *condiciones de extensión* son condiciones bajo las cuales el sistema toma diferente comportamiento. Decimos *extensión* en vez de *fallo* o *excepción* ya que incluimos escenarios alternativos como así también condiciones de fallo.

Aquí tenemos un ejemplo:

8. El sistema va a través del documento, verificando cada palabra con su diccionario de ortografía.
9. El sistema detecta un error de ortografía, resalta la palabra y presenta las alternativas al usuario.
10. El usuario selecciona una de las alternativas para reemplazarla. El sistema reemplaza la palabra resaltada con la seleccionada por el usuario.

Extensiones:

- 2.a. El sistema no detecta errores de ortografía hasta el fin del documento:
 - 2.a.1. El sistema notifica al usuario y termina el caso de uso.
- 3.a. El usuario elige mantener la palabra original:
 - 3.a.1. El sistema deja la palabra y continúa.
- 3.b. El usuario tipea una nueva palabra, que no está en la lista:
 - 3.b.1 El sistema vuelve a verificar la nueva palabra, vuelve al paso 1

6.1.1 Piense todos los fallos imaginables y cursos alternativos

Hemos encontrado importante pensar y obtener un buen panorama de las condiciones de extensión antes de sentarse a escribir el manejo de esas extensiones. Pensar una extensión y desarrollar correctamente cómo el sistema lo trata insume mucha energía. Probablemente se sentirá agotado después de tres o cuatro extensiones. Esto significa que no pensará el próximo conjunto de condiciones de extensión como debería.

Si por el contrario, primero piensa en todos los caminos alternativos y situaciones de fallo, obtendrá una lista que actuará como andamiaje de su trabajo en las próximas horas o días. En la etapa inicial, piense todos los posibles caminos en los cuales el escenario puede fallar, o cursos alternativos que pueden ocurrir. Asegúrese de considerar:

- Caminos alternativos (“El empleado usa una tecla rápida”)
- El actor primario comportándose incorrectamente (“Password inválido”)
- Inacción del actor primario (“Time out esperando por el password”)
- Cada ocurrencia de la frase “el sistema valida” implica que habrá una extensión para manejar la falla de la validación (“Número de cuenta inválido”)
- Falta de respuesta o inapropiada de un actor secundario (“Time out esperando respuesta”)
- Falla interna del sistema bajo diseño, que debe ser detectado y manejado como parte del proceso del negocio (“El cajero se atascó”)
- Falla interna inesperada y anormal, que debe ser manejada y una consecuencia externa visible (“Log de transacciones defectuoso descubierto”)
- Falla de performance crítica del sistema que debe ser detectado (“Respuesta no calculada en 5 segundos”)

Directriz 11: La condición indica que fue detectado.

Escriba lo que *el sistema detecta*, no lo que *pasó*. No escriba “El cliente se olvidó la clave”. El sistema no puede detectar que olvidó su clave. ¿Qué es lo que el sistema detecta en este caso? Inacción, lo que significa que el tiempo límite fue excedido. Escriba “Límite de tiempo excedido esperando por ingreso de clave”.

Las condiciones son usualmente frases describiendo que fue detectado.

Nos gusta poner dos puntos (‘:’) después de la condición para asegurarnos que el lector no pensará accidentalmente que es un paso. Si está usando pasos numerados, ponga el número del paso donde la condición podría ser detectada y luego una letra (Ej: 4.a.). No hay secuencia asociada a las letras, así que no hay implicancia de que 4.b sigue a 4.a. Esto nos permite adjuntar tantas condiciones de extensión como querramos para cada paso.

- 2.a. Fondos insuficientes:
- 2.b. La red se cayó:

Si la condición puede ocurrir en varios pasos y siente que es importante indicarlo, simplemente liste los pasos donde puede ocurrir:

- 2-5a. El usuario abandonó repentinamente:

Si la condición puede ocurrir en cualquier momento, use el asterisco (*) en lugar del número de paso. Liste las condiciones con asterisco antes de las numeradas.

***a. La red se cayó:**

2.a Fondos insuficientes

No se preocupe si el fallo ocurre en el paso cuando el usuario ingresa algún dato o en el paso posterior, cuando el sistema valida el dato. Uno podría ponerse a discutir que el error ocurre en cualquiera de los dos, pero no vale la pena perder el tiempo. Usualmente lo ponemos con el paso de la validación si hubiera.

Nota: La lista contendrá más ideas de la que finalmente se usarán. Eso está bien. El punto está en tratar de capturar todas las situaciones que el sistema encontrará en el caso de uso. Ya podrá reducirla más adelante.

6.1.2 Racionalizar la lista de extensiones

El propósito de la racionalización es reducir la lista lo máximo posible. La lista de extensiones ideal muestra todas las situaciones que el sistema debe manejar, y sólo ellas. Recordemos, siempre, que un documento de requerimientos largo es difícil de leer y de mantener. Combinando condiciones de extensión, se acorta la escritura y la lectura.

Después de la primera fase, tendrá un corto y simple escenario principal, y una larga lista de condiciones a considerar. Estudie la lista cuidadosamente, eliminando las que el sistema no necesita manejar, y combinando aquellas que tienen el mismo efecto sobre el sistema. Use estos dos criterios:

- El sistema debe poder detectar la condición
- El sistema está obligado a manejar la condición detectada

Pruebe convertir condiciones no detectables en detectables antes de eliminarlas. Si la condición es “El cliente olvidó su tarjeta”, elimínela, ya que no hay una condición equivalente que el sistema pueda detectar. Si la condición es “El cliente olvidó su clave”, no la elimine, conviértala en “Time out esperando ingreso de clave”, la cual el sistema puede detectar.

A continuación, combine condiciones equivalentes. Usted puede haber escrito estas tres condiciones: “La tarjeta está dañada”, “El lector de la tarjeta no funciona bien”, “La tarjeta no es de este cajero automático”. Desde un punto de vista de requerimientos, el comportamiento del cajero automático es el mismo: Devolver la tarjeta y notificar al cliente. Usted podría tratar de combinar estas condiciones. Si no puede encontrar una frase significativa para todas ellas, simplemente haga una lista. “Tarjeta no legible o desconocida por el cajero”.

6.1.3 Combinar fallos

Además de combinar condiciones que producen el mismo efecto, combine fallos de los casos de uso de más bajo nivel que tienen el mismo efecto en el caso de uso de mayor nivel. Esta combinación de fallos de bajo nivel es una de las formas de evitar tener una explosión de extensiones en los casos de uso de alto nivel.

Considere, como ejemplo, que está trabajando en el paquete “Consejero Financiero Personal”. Está escribiendo el caso de uso de usuario *Actualizar inversión*. Vamos a suponer que uno de los últimos pasos dice:

Caso de uso: Actualizar inversión

7. El usuario hace que PAF salve el trabajo

8.

Esta referencia llama al caso de uso *Salvar trabajo*, que contendrá las siguientes condiciones:

Caso de uso: Salvar trabajo

....

Extensiones:

3.a. El archivo ya existe (el usuario no quiere sobrescribir):...

3.b. Directorio no encontrado:...

4.a. Excede capacidad de almacenamiento:...

4.b. Archivo protegido contra escritura:...

Salvar el trabajo finaliza con éxito o fallo, volviendo la ejecución al paso 7 de *Actualizar inversión*. Si es exitoso continúa en el paso 8. Pero, si falla? ¿Qué deberíamos escribir para la extensión 7.a? Los lectores de *Actualizar inversión* no sabrán por qué fallo. Todos los fallos tendrán el mismo efecto, entonces escriba en *Actualizar inversión* sólo una extensión, describiendo qué pasa cuando *Salvar trabajo* falla.

Caso de uso: Actualizar inversión

...

El usuario hace que PAF salve el trabajo

...

Extensiones:

7.a. Falla *Salvar trabajo*:

7.a.1 ...lo que sea que pase...

Lo mejor de combinar fallos es que aún en los casos de uso de más alto nivel, los fallos son escritos en el vocabulario apropiado para el nivel. Aún los ejecutivos ocupados pueden tomarse tiempo para leerlos, porque el reporte de fallos está al mismo alto nivel.

6.2 Manejo de extensiones

En la situación más simple, hay sólo una simple secuencia de pasos para tratar la condición.

Sin embargo, en general, la extensión es todo un caso de uso. El disparador es la condición de extensión. La meta es también completar la meta del caso de uso, o reponerse del fallo encontrado. El cuerpo es un conjunto de pasos, y posiblemente extensiones a esos pasos.

La extensión puede terminar alcanzando o abandonando la meta, igual que un caso de uso. La similitud no es accidental, y prueba ser muy conveniente en extensiones complicadas.

Comience escribiendo el manejo de la condición con el paso que sigue a la condición que fue detectada. No necesita repetir la condición detectada. Continúe igual que cuando escribe el escenario principal. Use todas las directrices sobre nivel de meta, estilo de verbo, y oraciones discutidas con anterioridad. Siga escribiendo hasta que alcance un lugar donde pueda volver a unirse al escenario principal o el caso de uso falle. Típicamente, una extensión finaliza en una de estas formas:

- **El paso extendido ha sido arreglado y reemplazado. Al final del manejo de la extensión es como si el paso haya tenido éxito.**

...

3. El usuario activa la dirección del website

4. ...

Extensiones:

3.a. No hay dirección disponible:

3.a.1 El usuario busca un nuevo website

3.b...

- **El sistema le da al actor otra chance. Al final del manejo de la extensión, se vuelve al principio del mismo paso. Note que el sistema vuelve a validar el password en el siguiente ejemplo:**

2. ...

3. El usuario ingresa el password.

4. El sistema valida el password.

5. ...

Extensiones:

4.a. Password inválido:

4.a.1 El sistema notifica al usuario, y pide el password otra vez.

4.a.2 El usuario reingresa el password

4.b....

- **El caso de uso finaliza, a causa de un fallo total.**
 2. ...
 3. El usuario ingresa el password
 4. El sistema valida el password
 5. ...Extensiones:
...
 - 4.c Ingreso de password inválido demasiadas veces:
 - 4.c.1 El sistema notifica al usuario, y termina la sesión.
 - 5.a ...
- **El comportamiento sigue un camino completamente diferente para alcanzar la meta.**
 2. ...
 3. El usuario....
 4. El usuario....
 5. ...Extensiones:
 - 3.a. El usuario corre un macro personal para completar el procesamiento:
 - 3.a.1 termina el caso de uso

En los dos primeros casos, no es necesario decir qué pasa a continuación en la extensión, porque es obvio para el lector que el paso se reiniciará o continuará. En los otros dos, generalmente no es necesario decir más que “fallo” o “ el caso de uso termina”, porque los pasos muestran al sistema cuidando los intereses de las personas.

La mayoría de las extensiones no dicen donde se vuelve después de la extensión. Generalmente es obvio, y escribir “vuelve al paso N” después de cada extensión vuelve el texto más difícil de leer. En raras ocasiones es necesario a qué parte del escenario principal vuelve la extensión.

Directriz 12: El manejo de condiciones va indentado

Cuando use el estilo numerado mostrado en este texto, indente los pasos que muestran cómo la condición es manejada, y comience la numeración de nuevo desde 1, después de la letra. Los pasos siguen todas las directrices dadas anteriormente.

Extensiones:

2.a. Fondos insuficientes:

2.a.1 El sistema notifica al cliente, y pide una nueva suma.

2.a.2 El cliente ingresa nueva suma.

Cuando usamos el estilo de prosa directa (sin numerar), también indente o comience un nuevo párrafo para los pasos.

6.2.1 Fallos dentro de fallos

Dentro de una extensión, puede encontrar un nuevo punto de extensión, probablemente un fallo. Si está usando el estilo indentado de escritura, simplemente indente de nuevo, y continúe nombrando las condiciones y los escenarios como antes. En algún punto, su indentación y numeración será tan compleja, que decidirá abrir la extensión en otro caso de uso. Esto pasa sobre cerca del tercer nivel de indentación. He aquí un ejemplo:

6.a. El empleado decide salir sin completar la información mínima:

6.a.1 El sistema advierte al empleado que no puede salir y finalizar sin ingresar fecha, nombre o número de póliza

6.a.1.a El empleado elige continuar ingresando.

6.a.1.b El empleado salva como reporte intermedio y sale.

6.a.1.c El empleado insiste en salir sin ingresar información mínima:

6.a.1.c.1 El sistema deshace cualquier versión intermedia salvada y sale.

6.2.1 Creando un nuevo caso de uso de una extensión

Para abrir una extensión en un nuevo caso de uso, simplemente decida cuál es la meta del actor primario, dé al caso de uso un nombre, etiquételo con su nivel (probablemente subfunción), y complete los detalles del caso de uso.

Imagine un caso de uso *Administrar Reportes* que contiene un paso que dice “El usuario salva o imprime el reporte en cualquier momento”. Tiene un conjunto de extensiones describiendo varias alternativas y condiciones de fallo. Pero esa lista de extensiones continuará creciendo: reporte sin nombre, nombre ya existente (sobreescribir o no), usuario cancela en la mitad, y más. En definitiva, los escritores decidieron poner *Salvar reporte* en un caso de uso propio. En el caso de uso original, todavía deberá tratar con el hecho de que el nuevo sub caso de uso puede fallar, entonces su descripción será probablemente para mostrar el escenario principal y condiciones de fallo.

Desde un punto de vista teórico y teniendo en cuenta el esfuerzo, es simple poner una extensión en un caso de uso aparte o volver atrás. El modelo de casos de uso nos permite considerar a esta última una decisión menor. No hay problema en mover fuera la extensión *Salvar reporte*, y de igual forma, toma unos pocos minutos con el editor de texto moverlo otra vez dentro de *Administrar reportes*.

Sin embargo, el costo de crear un caso de uso no está en el esfuerzo necesario para escribirlo. El nuevo caso de uso debe ser etiquetado, rastreado, agendado, testeado y mantenido. Esas son operaciones costosas para el equipo del proyecto.

Mantener una extensión dentro del caso de uso generalmente tiene económicamente más sentido. Dos situaciones que llevarán a crear un nuevo caso de uso para la extensión:

- La extensión es usada en varios lugares. Poniendo la extensión en su propio caso de uso se puede seguir y mantener en un solo lugar. Idealmente, esta es la única razón para crear un caso de uso bajo el nivel del mar.
- La extensión hace al caso de uso realmente difícil de leer. El límite de legibilidad es alrededor de 2 páginas por caso de uso, y tres niveles de indentación.