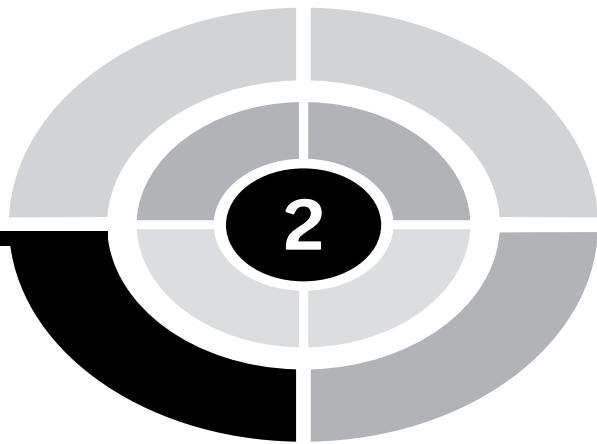


CAPÍTULO



El principio con casos de uso

El Unified Modeling Language (UML) soporta el análisis y diseño orientados a objetos proporcionándole una manera de captar los resultados del análisis y el diseño. En general, iniciamos con la comprensión de nuestro problema; es decir, el análisis. Un tipo excelente de modelo para captar el análisis es el diagrama de casos de uso.

La finalidad de un *caso de uso* es describir la manera en que se usará un sistema: describir sus finalidades esenciales. La finalidad de los *diagramas* de casos de uso es captar en forma visual las finalidades esenciales.

Un caso de uso bien escrito y bien representado en diagrama es una de las clasificaciones de modelos individuales más importantes que usted puede crear. Esto es así porque expresar con claridad, conocer y organizar los objetivos es singularmente importante para alcanzarlos con éxito. Existe un viejo proverbio que dice: “Un viaje de mil millas empieza con un paso”, y existe un proverbio un poco menos antiguo que dice: “Si no sabe hacia adónde va, entonces el viaje nunca terminará.”

En este capítulo, hablaré acerca de una primera parte significativa de ese viaje —la creación de casos de uso— que cubrirá

- Los símbolos usados para crear los diagramas de casos de uso
- Cómo crear los diagramas de casos de uso
- Cuántos diagramas de casos de uso crear
- Cuánto incluir en un diagrama de casos de uso
- El nivel de detalle a incluir en un diagrama de casos de uso
- Cómo expresar las relaciones entre los casos de uso individuales
- La cantidad y el estilo de texto que es útil para hacer anotaciones en los diagramas de casos de uso
- De manera significativa, cómo establecer las prioridades de los casos de uso

Cómo hacer el caso para las casos de uso

Los diagramas de casos de uso parecen muy fáciles; constan de figuras de línea, líneas y óvalos. La figura de palillos se llama *actor* y representa a alguien o algo que actúa sobre el sistema. En el desarrollo de software, los actores son personas u otro software que actúa sobre el sistema. Las líneas son punteadas o continuas, con varias flechas o sin ellas, que indican la relación entre el actor y los óvalos. Estos últimos son los casos de uso y, en el diagrama de casos de uso, los óvalos tienen algún texto que proporciona una descripción básica. La figura 2-1 es un ejemplo sencillo de un diagrama de casos de uso.

Durante mucho tiempo los diagramas de casos de uso me fastidieron porque parecían demasiado sencillos para tener algún valor. Un niño de tres o cuatro años con un crayón y un pedazo de papel podría reproducir estas figuras de línea; sin embargo, su sencillez es decepcionante.

Que un diagrama de casos de uso sea fácil de crear es un elogio implícito para el UML. Hallar los casos de uso correctos y registrar sus responsabilidades en forma correcta es la decepción. Hallar los casos de uso correctos y describirlos de manera adecuada es el proceso crítico que impide que los listos ingenieros de software pasen por alto necesidades

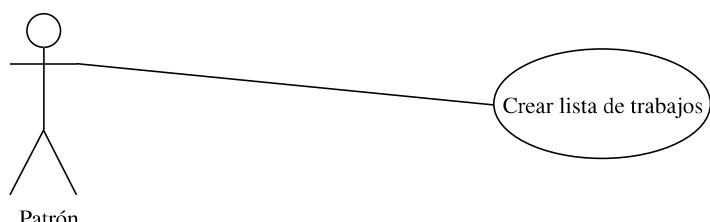


Figura 2-1 Un diagrama de casos de uso muy sencillo.

críticas y que inventen de manera innecesaria. En pocas palabras, los diagramas de casos de uso constituyen un macrorregistro de lo que usted quiere estructurar.

En el párrafo anterior, usé el prefijo *macro*. Macro en este contexto sencillamente significa “grande”. Los grandes objetivos, o macroobjetivos, son los que se mencionan como los argumentos, o razones, poderosos de la empresa para hacer algo. En los diagramas de casos de uso se captan los objetivos grandes, poderosos. En el texto de esos casos se captan los detalles de apoyo.

Esto es lo que se me escapaba en las imágenes de las figuras de línea de los diagramas de casos de uso; perdía de vista que, sencillamente, al registrar lo que el sistema hará y lo que no hará, registramos y especificamos el alcance de lo que se está creando; también perdía de vista que el texto que acompaña los diagramas de casos de uso rellena los espacios en blanco entre los macroúsos y los microúsos, en donde *micro* significa usos “menores, de apoyo”.

Además de registrar los usos primarios y secundarios, los diagramas de casos de uso nos proporcionan en forma implícita varias oportunidades significativas para administrar el desarrollo, a lo cual entrará con más detalle a medida que avance el capítulo.

Establecimiento de prioridad de las capacidades

¿Alguna vez ha escrito una lista de cosas por hacer? Una *lista de cosas* por hacer es una lista de cosas que usted debe hacer o desea hacer. El acto de escribir la lista es un punto de partida. En esencia, los casos de uso son listas de cosas por hacer. Una vez que ha captado los casos de uso, ha articulado lo que el sistema hará, y puede usar la lista para dar prioridades a nuestras tareas. Tanto enunciar como organizar los objetivos son primeras tareas muy críticas.

El valor de establecer prioridades para las capacidades de un sistema es que el software es fluido. Permítame ilustrar, por medio de un ejemplo, lo que quiero decir. Es posible crear, guardar, abrir e imprimir un documento de texto tanto con Notepad (Bloc de notas) como con Word de Microsoft, pero la diferencia en el número de líneas de código y el número de características entre estos dos programas es tremenda. Al establecer prioridades de los usos, con frecuencia tenemos la oportunidad de hacer, con ventaja, malabarismos con las características, el presupuesto y el programa.

Suponga, por ejemplo, que mis objetivos primarios son ser capaz de crear, guardar, abrir e imprimir un documento de texto. Suponga además que mis objetivos secundarios son guardar el documento como texto llano, HyperText Markup Language (HTML, lenguaje de marcado de hipertexto), y como texto enriquecido, es decir, formateo especial. Establecer prioridades de las capacidades significa que podría elegir enfocarme hacia los usos primarios —crear, guardar, abrir e imprimir—, pero aplazar el soporte de HTML y texto enriquecido. (Las características en el software por lo común se aplazan hacia versiones posteriores, debido a las restricciones reales mencionadas con anterioridad, incluyendo tiempo, presupuesto y un cambio en el entorno de la empresa.)

No tener tiempo suficiente y quedarse sin dinero son problemas directos. Los desarrolladores de software son rutinariamente optimistas, se distraen en salidas por la tangente y pasan más tiempo en reuniones que en la planeación, y estas cosas gravan un presupuesto. Sin embargo, tomemos un momento para examinar un cambio en el entorno de la empresa. Si nuestras necesidades originales fueron HTML, texto llano y texto enriquecido y hemos estado estructurando nuestro software en los últimos cinco años, resultaría perfectamente plausible que un cliente dijera, a la mitad del curso del desarrollo, que guardar un documento como eXtensible Markup Language (XML, lenguaje ampliable de marcado) sería más valioso que como texto enriquecido. De este modo, debido a un clima tecnológico en evolución, a media corriente un cliente podría restablecer las prioridades y demandar XML como más importante que el texto enriquecido. Si no hubiéramos documentado nuestras necesidades primarias y secundarias, entonces podría ser un reto muy grande determinar los trueques deseables, como cambiar el texto enriquecido por el XML. Debido a que registramos con claridad los casos deseables de usos, podemos establecer prioridades y hacer trueques valiosos, si es necesario.

Comunicación con los no tecnófilos

Otra cosa que no perdí de vista acerca de los casos de uso es que su mera sencillez los hace un medio fácil de transmisión para comunicarse con no tecnófilos. A estas personas las llamamos *usuarios* o *clientes*.

Los programadores que usan el hemisferio izquierdo de su cerebro en general detestan a los usuarios. La idea básica es que si uno no puede leer el código, entonces es tonto o, por lo menos, más tonto que aquellos que sí pueden. El UML y los casos de uso cubren la brecha entre los programadores que usan el hemisferio izquierdo del cerebro y los usuarios no tecnófilos.

Una figura de palillos, una línea y un óvalo son suficientemente simplistas, cuando se combinan con algún texto, para que todos los participantes puedan entender el significado. El resultado es que los usuarios y clientes pueden observar los dibujos y leer el texto llano, y determinar si los tecnólogos han, o no, registrado con exactitud y comprendido las características deseables. Esto también significa que los administradores —quienes pueden no haber escrito código en 10 años— y las direcciones técnicas pueden examinar el producto final y, por inspección, garantizar que la inventiva desenfrenada no es la causa de los programas no cumplidos ni de las características ausentes. Demostrando esta disonancia al continuar con mi primer ejemplo, suponga que, de cualquier manera, se implementa el soporte de texto enriquecido porque el programador sabe cómo almacenar y recuperar ese tipo de texto. No obstante, debido a que el XML es más reciente y el programador tiene menos experiencia en trabajar con él, la característica de escritura en XML se aplaza sin maldicia. Un administrador proactivo puede descubrir las necesidades de un cliente, según se captan mediante los casos de uso, y apropiarse de salidas por la tangente improductivas.

Debido a que los casos de uso son visuales y sencillos, los usuarios y clientes pueden suministrar retroalimentación, y las personas que constituyen el puente entre los clientes y los programadores, como los administradores, pueden determinar si las características que en realidad se estructuraron reflejan con exactitud los deseos de los usuarios.

Uso de los símbolos de los casos de uso

Los diagramas básicos de casos de uso constan de sólo unos cuantos símbolos: el *actor*, un *conector* y el *óvalo del caso de uso* (figura 2-2). Tomemos unos cuantos minutos para hablar de cómo se usan estos símbolos y qué información transmiten.

Símbolos de actores

La figura de palillos, mencionada como *actor*, representa participantes en los casos de uso. Los actores pueden ser personas o cosas. Si un actor es una persona, entonces, en realidad, nunca se puede representar por medio de un código. Si un actor es otro subsistema, entonces se le puede observar como una clase o subprograma, pero todavía representarse usando el símbolo de actor en los diagramas de casos de uso.

Los actores se descubren como resultado del análisis. Conforme vaya identificando los macrourgos del sistema, identificará quiénes son los participantes para esos casos de uso. En principio, registre cada actor a medida que se descubre, agregando un símbolo de actor a su modelo y describiendo cuál es su papel. Nos preocuparemos acerca de la organización y el refinamiento más adelante, en la sección titulada “Creación de los diagramas de casos de uso”.

Casos de uso

El símbolo del caso de uso se utiliza para representar capacidades. Al caso de uso se le da un nombre y una descripción mediante un texto. Este último debe describir cómo inicia y finaliza el caso de uso, e incluye una descripción de la capacidad descrita por el nombre de la misma, así como escenarios de apoyo y requisitos no funcionales. En la sección titulada “Creación de los diagramas de casos de uso”, examinaremos ejemplos

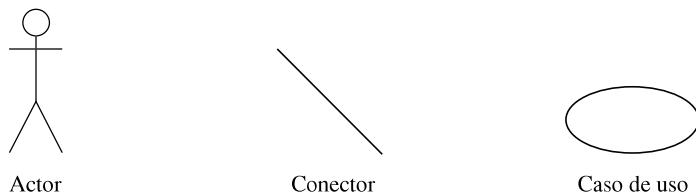


Figura 2-2 Los símbolos básicos de los diagramas incluyen al actor, al conector y al óvalo del caso de uso.

de nombres de casos de uso, y en la sección titulada “Documentación de un caso de uso utilizando un borrador”, proporcionaré un borrador modelo que pueda usar para ayudarse a escribir las descripciones de los casos de uso.

Conectores

Dado que los diagramas de casos de uso tienen múltiples actores y en virtud de que los casos de uso pueden estar asociados con los actores y con otros casos de uso, se utilizan los conectores para indicar la manera en que ambos están asociados. Además, los estilos de conectores pueden cambiar para transmitir más información acerca de la relación entre los actores y los casos de uso. Por último, los conectores pueden tener adornos y anotaciones que suministran incluso más información.

Estilos de líneas para los conectores

Existen tres estilos básicos de líneas para los conectores. Un conector de línea simple se llama *asociación* y se usa para mostrar cuáles actores están relacionados con cuáles casos de uso. Por ejemplo, en la figura 2-1 se mostró que un patrón está asociado con el caso de uso “Crear lista de trabajos”.

Un segundo estilo de conector es una línea punteada con una flecha direccional (figura 2-3). Este estilo de conector se conoce como *dependencia*. La flecha apunta hacia el caso de uso del que depende. Por ejemplo, suponga que a los patrones de www.motown-jobs.com se les debe dar acceso para crear una lista de trabajos. Entonces podemos decir que el caso de uso “Crear lista de trabajo” depende de un caso de uso “Entrar”. Ésta es la relación que se ilustra en la figura 2-3.

Un tercer estilo de conector es una línea dirigida con un triángulo hueco, al cual se le conoce como *generalización*. La palabra *generalización* en el UML significa “herencia”. Cuando mostramos una relación de generalización entre dos actores o dos casos de uso, estamos indicando que el actor o el caso de uso “hijos” son un caso del actor o uso básico y algo más. En la figura 2-4, se muestra una relación de generalización entre dos actores y dos casos de uso.



Figura 2-3 El caso de uso “Crear lista de trabajos” depende de que el patrón obtenga acceso.

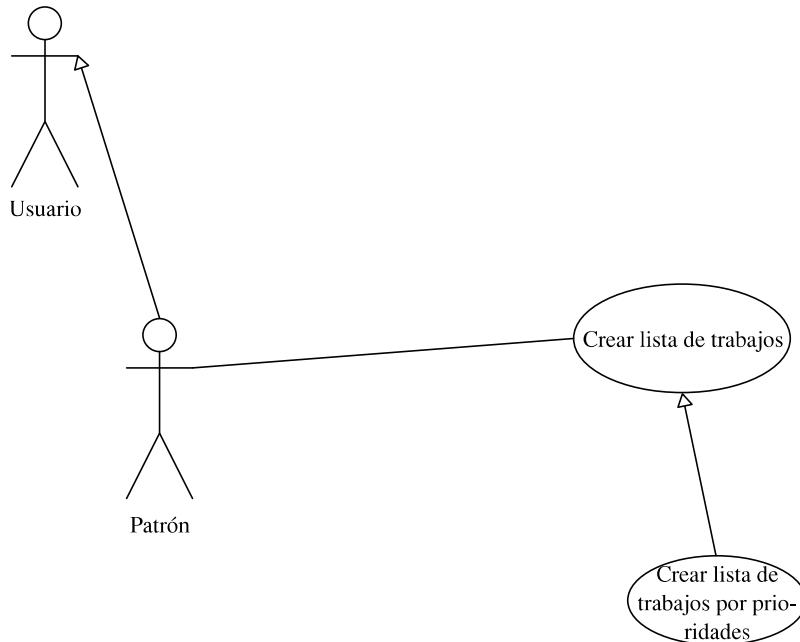


Figura 2-4 Diagrama de casos de uso en el que se muestran dos relaciones de generalización entre dos actores y entre dos casos de uso.

En las relaciones de generalización, la flecha apunta hacia la cosa sobre la cual nos estamos expandiendo. Existen varias maneras en las que usted puede describir esta relación en forma verbal —acerca de la cual usted debe saber—, pero desafortunadamente, todos estos sinónimos pueden conducir a confusión verbal. Los siguientes enunciados describen las relaciones de generalización que se muestran en la figura 2-4:

- El usuario es el objetivo y el patrón es la fuente.
- El patrón es un usuario.
- El usuario es el subtipo y el patrón es el supertipo.
- El patrón se hereda del usuario.
- El usuario es el tipo padre y el patrón es el tipo hijo.
- El patrón generaliza al usuario.

(En esta lista, puede sustituir la frase *Crear lista de trabajos* en todas partes en donde vea la palabra *Usuario*, y sustituir la frase *Crear lista de trabajos por prioridades* en todas las partes en donde vea la palabra *Patrón*, para transmitir la relación entre los dos casos de uso.) El último enunciado, en el cual se usa la palabra *generaliza*, es el más exacto en el contexto del UML, pero vale la pena reconocer que todos los enunciados son equivalentes.

Adornos de los conectores

Los diagramas UML fomentan el uso de menos texto porque las imágenes transmiten una gran cantidad de información a través de una conveniente taquigrafía visual, pero los diagramas UML no se abstienen por completo del texto; por ejemplo, los conectores pueden incluir texto que indique multiplicidad de los puntos extremos y texto que estereotipa el conector.

Manera de mostrar la multiplicidad

En general, los conectores pueden tener notaciones de multiplicidad en cualquiera de sus dos extremos. Las notaciones de multiplicidad indican el conteo posible de cada cosa. Por ejemplo, un asterisco significa muchos; un asterisco próximo a un actor significa que puede haber muchos ejemplos de ese actor. Aun cuando el UML permite hacer anotaciones de esta manera en los conectores de caso de uso, eso no es muy común. Es más probable que usted vea estas marcas de notación de conteo en diagramas del tipo de los de clase, de modo que daré detalles sobre la multiplicidad en el capítulo 3.

Estereotipado de los conectores

Una notación más común en los conectores es el estereotipo. Los estereotipos agregan detalles a la relación entre los elementos en un diagrama de caso de uso. Por ejemplo, en la figura 2-3, introduce el conector de dependencia. Se puede usar un estereotipo para ampliar el significado de este conector.

En la sección titulada “Estilos de líneas para los conectores”, dije que un patrón puede crear una lista de trabajos e ilustré esto con un actor patrón, un caso de uso “Crear lista de trabajos” y un conector de asociación; sin embargo, también dije que el patrón debe obtener acceso. Cuando un caso de uso —“Crear lista de trabajos”— necesita los servicios de otro caso de uso —“Entrar”— entonces se dice que el caso de uso dependiente *incluye* el caso de uso del que depende. (En código, una relación incluir se implementa como reutilización de código.)

Un estereotipo se muestra como texto entre los caracteres « y » (comillas angulares). Por ejemplo, si decimos que “Crear lista de trabajos” incluye a “Entrar”, entonces podemos representar un estereotipo *incluir* colocando una anotación en el conector de dependencia como se muestra en la figura 2-5.



Figura 2-5 Ejemplo de un estereotipo *incluir*—usado para representar *reutilizar*—en la dependencia entre “Crear lista de trabajos” y “Entrar”.

Incluir y extender son conceptos importantes en los diagramas de caso de uso, de modo que enseguida ampliaré lo relativo a estos temas.

Nota *Estereotipo es un concepto generalmente útil en el UML. La razón de esto es que es permisible que usted introduzca y defina sus propios estereotipos. De esta manera, puede extender el UML.*

Caso de uso de inclusión y de extensión

Una relación de dependencia entre dos casos de uso significa que, de alguna manera, el caso dependiente necesita al caso del que depende. Dos estereotipos de uso común y predefinidos que refinan las dependencias en los casos de uso son el incluir y el extender. Tomemos un minuto para ampliar nuestros comentarios de introducción sobre incluir, de la sección anterior, e introduzcamos extender.

SUGERENCIA *Visio aplica un estereotipo extender en el conector de generalización para dar a entender herencia. Existen variaciones entre el UML y las herramientas del mismo, porque el UML es un estándar en evolución y la implementación de las herramientas puede ir adelante o atrás de la definición oficial del UML.*

Más sobre los estereotipos incluir

Una dependencia rotulada con el estereotipo incluir significa que, finalmente, el caso de uso dependiente es para volver a usar el caso del que depende. El equipaje que va con el estereotipo incluir es que el caso de uso dependiente necesitará los servicios del caso del que depende y saber algo acerca de la realización de ésta, pero lo opuesto no es cierto. El caso de uso del que se depende es una entidad completa y distinta que no debe depender del caso dependiente. La concesión de acceso es un buen ejemplo. Resulta claro que requerimos que un patrón tenga acceso para crear una lista de trabajos, pero también pudimos obtener acceso por otras razones.

Nota *En una dependencia incluir entre casos de uso, el caso dependiente también se conoce como el caso de uso básico, y aquella de la que se depende también se conoce como el caso de uso de inclusión. Aunque básico y de inclusión pueden ser términos más precisos, no parece que se empleen de manera común al hablar.*

Poner tanto significado en una pequeña palabra como *incluir* es la razón por la cual el UML puede transmitir una gran cantidad de significado en un diagrama sencillo, pero también es la razón por la cual los modelos UML pueden representar un reto para crearse y leerse. Una estrategia real a la que puede recurrir es agregar una nota en donde no esté

seguro acerca del uso de algún aspecto idiomático del UML (vea, más adelante, “Anotaciones en los diagramas de caso de uso”). Por ejemplo, si quiere describir la relación entre “Crear lista de trabajos” y “Entrar”, pero no está seguro acerca de cuál conector o cuál estereotipo usar, entonces podría usar una asociación simple y una nota asociada al conector que describa en texto llano lo que usted quiere dar a entender. La nota puede actuar como un recordatorio para, más adelante, regresar y buscar el UML preciso.

Uso de los estereotipos extender

El estereotipo extender se usa para agregar más detalle a una dependencia, lo cual significa que estamos agregando más capacidades (como ejemplo, vea la figura 2-6). Como se muestra en la figura, decimos que “Registrar las listas vistas” extiende (y depende de) “Ver lista”.

NOTA En una relación extendida, la flecha apunta hacia el caso de uso básico y el otro extremo se conoce como el caso de uso de extensión.

En la sección anterior, no permitiríamos que un patrón creara una lista de trabajos sin registrarse, pero en el caso del registro del caso de uso entrar es indiferente para la reutilización del caso. En esta sección, a el caso de uso ver lista no le importa que la estén registrando; en otras palabras, la característica registrar necesitará saber acerca de la característica ver lista, pero no en sentido contrario.

Una perspectiva valiosa aquí es quién podría estar interesado en el registro. Es evidente que al “Solicitante de trabajo” tal vez no le interese cuántas veces se ha visto la lista, pero un patrón previsor podría estar interesado en cuánto tráfico está generando su lista. Pasemos ahora por un momento a un dominio diferente. Suponga que el “Solicitante de trabajo” fuera el comprador de una casa y que la lista lo fuera de residencias. Ahora tanto el comprador como el vendedor podrían estar interesados en el número de veces que se ha visto la propiedad. Una casa que ha estado en el mercado durante meses puede tener

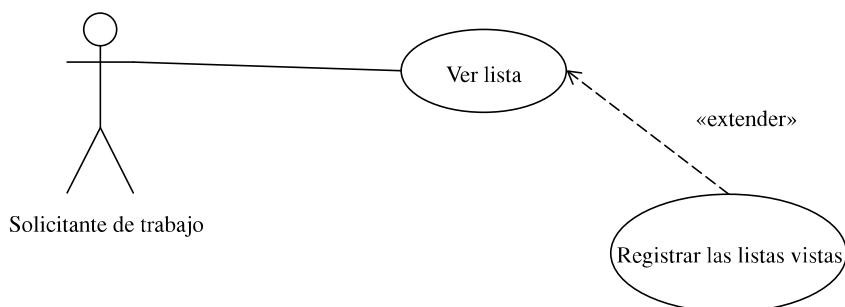


Figura 2-6 Seguir el rastro del número de veces que se ve una lista de trabajos es una extensión de “Ver lista”, como se describe mediante la dependencia y el estereotipo extender.

problemas. Sin embargo, en los dos escenarios, la lista es lo más importante y el número de veces que se ha visto es secundario. Esto ilustra la noción de caso de uso de extensión como parecidas a las características y, desde una perspectiva de mercadeo, las extensiones podrían ser elementos que estén separados en un paquete opcional de características.

SUGERENCIA Consideré la alternativa, ya que se relaciona con un caso de uso de extensión. Los casos de uso de extensión son características secundarias naturales. Si su proyecto tiene un programa apretado, lleve hasta el final los casos de uso de extensión, y, si su tiempo se agota, entonces posponga los casos de uso de extensión para una versión posterior.

Incluir y extender parecen algo semejantes, pero la mejor manera de tenerlos en orden es recordar que “la relación incluir es para volver a aplicar el comportamiento modelado por otro caso de uso, en tanto que la relación extender es para agregar partes a caso de uso existentes así como para modelar servicios opcionales del sistema” (Övergaard y Palmkvist, 2005, p. 79).

Anotaciones en los diagramas de casos de uso

Considere el trabajo de un estenógrafo en un juicio. Los estenógrafos usan esas graciosas máquinas estenográficas de escribir que producen una suerte de majaderías taquigráficas. Podemos suponer con seguridad que si una máquina de escribir común o un procesador de palabras pudiera aceptar una entrada suficientemente rápida como para mantenerse al ritmo del habla natural, entonces el estenógrafo nunca se hubiera inventado.

Los estenógrafos producen una taquigrafía que es más condensada que el discurso al hablar. El UML es como la taquigrafía para el código y el texto, y las herramientas de modelado del UML son semejantes a los estenógrafos. La idea es que los modelos se puedan crear más rápido que el código o más rápido que escribir descripciones en forma de texto. Dicho eso, a veces no hay un buen sustituto para el texto.

Si se encuentra en el predicamento de que sólo el texto parece resolver —o no está seguro del UML—, entonces siga adelante y agregue texto. Puede agregar texto mediante la documentación de sus modelos con características de la mayoría de las herramientas de modelado, agregando referencias URL a los documentos más verbosos o agregando notas directamente en los propios diagramas. Sin embargo, si agrega demasiado texto, entonces de manera natural, tardará más en completar el modelado y puede ser que se requiera un esfuerzo mayor para entender el significado de cada uno de los diagramas.

Inserción de notas

El UML es una taquigrafía para una gran cantidad de texto y de código, pero si lo necesita, siempre puede agregar texto. Todos los diagramas, incluyendo los casos de uso, permiten

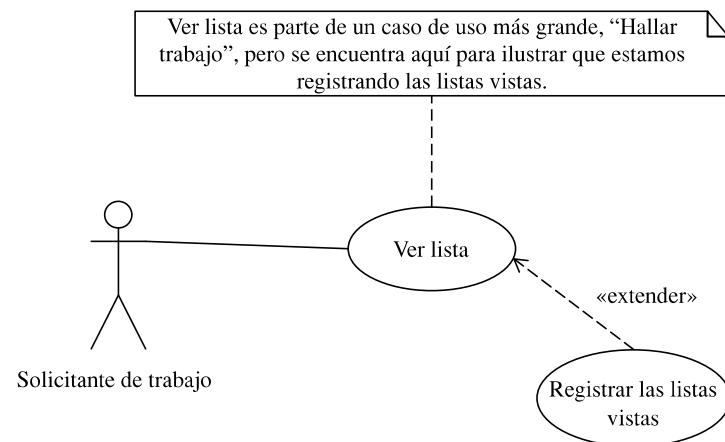


Figura 2-7 Nota que agrega texto llano para aclarar algún aspecto de un diagrama.

ten que se les agreguen anotaciones en forma de texto. Las notas se representan como un trozo de papel con una punta doblada y una línea que une el cuadro de texto al elemento que se le está haciendo la anotación (figura 2-7). Use las notas con moderación, porque pueden abarrotar un diagrama y hacerlo difícil de leer.

Modo de agregar documentación de soporte

Todas las herramientas de modelado que he usado —Together, Rose, Rose XDE, Visio, Poseidon para UML y la de Cayenne Software— permiten la documentación del modelo. Por lo común, esta documentación toma dos formas: texto que se almacena en el modelo y Uniform Resource Locators (URL, localizadores uniformes de recursos), que hacen referencia a documentos externos (figura 2-8). El examen de las características de su herramienta particular le descubrirá estas capacidades.

Más importante es qué tipo de documentación debe usted proporcionar. De manera subjetiva, la respuesta es tan pequeña como aquella con la que pueda ponerse en marcha, pero en general, los diagramas de caso de uso parecen necesitar lo máximo.

Los diagramas de caso de uso son bastante básicos con sus figuras de línea, pero son bastante importantes porque registran las capacidades que tendrá el sistema de usted. Una buena información a incluir con sus diagramas de caso de uso es

- Un párrafo conciso en el que se describa cómo empieza el uso, incluyendo cualesquiera condiciones previas
- Un párrafo corto para cada una de las funciones primarias
- Un párrafo corto para cada una de las funciones secundarias
- Un párrafo corto para cada uno de los escenarios primarios y secundarios, los cuales ayuden a ubicar en un contexto la necesidad de las funciones

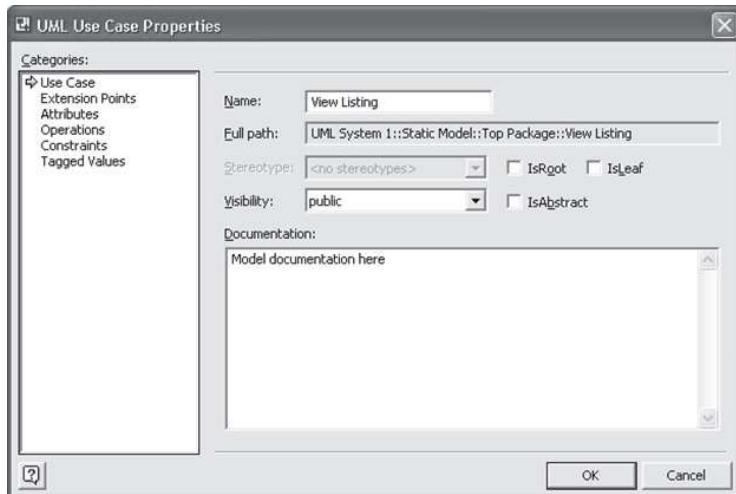


Figura 2-8 Mediante un doble clic sobre un elemento de modelo en Visio, puede añadir documentación que está agregada en el modelo.

- Un párrafo para las necesidades no funcionales
- Puntos de inserción en donde se usen cualesquiera otros casos de uso dependientes
- Un punto de finalización con las condiciones posteriores

Todos estos elementos suenan como una gran cantidad de trabajo, y pueden serlo. Sin embargo, recuerde que los casos de uso son los fundamentos del análisis, y es importante que las documente tan cuidadosa y completamente como pueda. De igual importancia es notar que usé las palabras *conciso* y *corto* de manera intencional. Por corto, quiero dar a entender que es aceptable tener párrafos de una sola oración.

Puede usar cualquier formato que le guste para documentar sus casos de uso. Si se siente cómodo con el formato de borrador, es muy fácil crear un borrador modelo con base en la lista con viñetas que acabo de dar. Una buena práctica es elegir un estilo para su documentación y adherirse a él.

Tomemos un momento para explicar en detalle los elementos —según se describen en la anterior lista con viñetas— de la documentación del caso de uso. Tenga presente que esto no es una ciencia exacta y que su documentación de los casos de uso no necesita ser perfecta.

Documentación de un caso de uso utilizando un borrador

Puede usar texto de forma libre para documentar un caso de uso, pero encuentro que un modelo de borrador sugiere la extensión de la información y actúa como un recordatorio de los elementos necesarios para documentar cada caso en forma adecuada. A continua-

ción se presenta un modelo que incluye una breve descripción y un ejemplo para cada sección. Vale la pena hacer notar que este estilo de documentación no es parte del UML, pero es un elemento útil del modelado.

1. Título

- a. Descripción: Use aquí el nombre del caso de uso, pues facilita mucho el acoplamiento de los diagramas de caso de uso con su documentación respectiva.
- b. Ejemplo: Mantener lista de trabajos.

2. Inicios del caso de uso

- a. Descripción: Describa con brevedad las circunstancias que llevan al caso de uso, incluyendo las condiciones previas. Deje fuera los detalles de la implementación, como “El usuario hace clic en un hipervínculo”, o las referencias a las formas, los controles o los detalles específicos de la implementación.
- b. Ejemplo: Este caso de uso se inicia cuando un patrón, un agente de un patrón o el sistema quiere crear, modificar o eliminar una lista de trabajos.

3. Funciones primarias

- a. Descripción: Los casos de uso no son necesariamente singulares. Por ejemplo, “Administrar la lista de trabajos” es un caso de uso razonable y puede incluir funciones primarias como leer un recipiente o escribir en él. La clave aquí es evitar demasiado pocas o demasiadas funciones primarias. Si necesita una buena medida, podrían ser dos o tres funciones primarias por caso de uso.
- b. Ejemplo: “CLAB la lista de trabajos.” Las funciones primarias de “Mantener la lista de trabajos” son crear, leer, actualizar y borrar la lista de trabajos.

4. Funciones secundarias

- a. Descripción: Las funciones secundarias son como un reparto de apoyo en una pieza teatral. Por ejemplo, dado un caso de uso “Administrar la lista de trabajos”, actualizar, insertar, crear y borrar una lista de trabajos —llamado *CLAB* por crear, leer, actualizar y borrar— son excelentes funciones secundarias, parte de un caso de uso más grande. Si necesita una medida, entonces el doble de funciones secundarias que de primarias es bueno.

b. Ejemplos:

- 1) “Hacer caducar la lista de trabajos.” Treinta días después de que la lista de trabajos se pone a disposición para ser vista, se dice que caduca. Una lista que ha caducado no se borra, pero es posible que los usuarios, con excepción del propietario de la lista, ya no puedan verla.
- 2) “Renovar la lista de trabajos.” Se puede extender una lista por 30 días adicionales mediante el pago de una tarifa adicional.

- 3) “Hacer que una lista de trabajos sea prioritaria.” En cualquier momento, durante la vida de una lista, su propietario puede elegir promoverla hacia lista prioritaria, mediante el pago de una tarifa prorrataeada por la parte consumida del periodo de la misma.
- 4) “Registrar las listas vistas.” Cada vez que se ve una lista, se escribirá una entrada de registro, haciendo constar la fecha y la hora en que se vio la lista y el protocolo de Internet (IP) del visitante.
- 5) “Examinar registros de las veces que se ha visto la lista.” En cualquier momento, el propietario puede ver la información registrada en relación con sus listas.
- 6) “Notificación automática de los registros de la vista de la lista.” El propietario de una lista de trabajos puede elegir que se le envíen por correo electrónico los registros de las vistas de esa lista, con un intervalo especificado por él.
- 7) “Pagar por la lista.” Se pide al propietario que pague por cada lista, a menos que ésta se ofrezca como un premio de promoción.

5. Escenarios primarios

- a. Descripción y ejemplo: Un escenario es un relato corto que describe las funciones en un contexto. Por ejemplo, dada una función primaria “Crear lista de trabajos”, podríamos escribir un escenario como éste: “La secretaria del Sr. García está por jubilarse, y él necesita contratar a alguien que la reemplace. Al Sr. García le gustaría una secretaria que mecanografe 100 palabras por minuto, quiera trabajar sólo cuatro horas al día y cobrar 10 dólares por hora. Necesita que la secretaria de reemplazo empiece a trabajar no después del 15 de enero.” Considere por lo menos tantos escenarios primarios como funciones primarias tenga. También considere un par de variaciones del escenario para las funciones importantes. Esto ayudará a que piense acerca de su problema en formas creativas. Hacer una lista de los escenarios en aproximadamente el mismo orden que el de las funciones que describe el escenario es una práctica útil.

6. Escenarios secundarios

- a. Descripción y ejemplo: Los escenarios secundarios son relatos cortos que ponen a las funciones secundarias en un contexto. Considere un escenario secundario al que nos referiremos como “Hacer caducar lista de trabajos”. Demostrado como un escenario, podríamos escribir: “El Sr. García pagó para que la lista se publicara durante 30 días. Después de 30 días, la lista de trabajos se retirará y se notificará al Sr. García por correo electrónico, dándole oportunidad de renovarla.” Podemos organizar los escenarios secundarios en un orden coherente con las funciones secundarias que apoyan.

7. Necesidades no funcionales

- a. Descripción: Las necesidades no funcionales se encargan de comportamientos implícitos, como con qué rapidez sucede algo o cuántos datos se pueden transmitir.
- b. Ejemplo: Debe procesarse el pago de un patrón en un periodo no mayor a 60 segundos, en tanto que él o ella, espera.

8. Finalizaciones de los casos de uso

- a. Descripción: En esta parte se describe lo que significa haber finalizado para el caso de uso.
- b. Ejemplo: El caso de uso ha finalizado cuando los cambios hechos en la lista de trabajos se han mantenido y se ha hecho el pago.

Cuánta información incluya en la parte escrita de sus casos de uso en realidad es decisión de usted. El UML guarda silencio acerca de este asunto, pero un proceso como el RUP le puede ofrecer alguna guía sobre el contenido, cantidad y estilo de la documentación en texto.

Como nota final, resulta útil registrar ideas acerca de las funciones y escenarios, incluso si finalmente elige descartarlos. Por ejemplo, podríamos agregar una función secundaria que exprese que “El sistema permitirá una renovación semiautomática de una lista de trabajos que están caducando”, apoyada por el escenario “La lista del Sr. García para contratar una nueva secretaría está próxima a caducar. Se notifica por correo electrónico al Sr. García que su lista está próxima a caducar. Al hacer clic sobre un vínculo en el correo, la lista del Sr. García se renueva en forma automática usando la misma facturación e información de pago usadas con la lista original”.

Al registrar y conservar las ideas consideradas, es posible hacer un registro de las ideas que se consideraron, pero puede ser que se lleven a cabo o que jamás se haga. Conservar un registro de las posibilidades impide que usted repita una y otra vez las ideas conforme los miembros del equipo vienen y se van.

Por último, resulta útil insertar referencias a los casos de uso del que se depende. En lugar de, por ejemplo, repetir un caso de uso de inclusión, sencillamente haga una referencia a ese caso en el punto en que se necesite. Por ejemplo, suponga que pagar por una lista de trabajos requiere que un patrón tenga acceso; en lugar de repetir el caso de uso “Entrar”, sencillamente hacemos una referencia a ella en donde se necesite; en este caso, podemos hacer una referencia a “Entrar” cuando hablamos acerca de pagar por la lista de trabajos.

Creación de los diagramas de casos de uso

Como mencioné al principio, los casos de uso son listas de diseños por hacer. Ya que un día feriado siempre está precisamente a la vuelta de la esquina, una buena analogía

comparativa es que definir casos de uso es como escribir una lista de tareas en orden para preparar su casa para una gran visita de parientes. Por ejemplo, podría escribir “Desempolvar la sala”. Entonces decide que su hija de 10 años hizo un buen trabajo la última vez, de modo que le pide a ella que desempolve. En este caso, el nivel de detalle es importante, porque sabe —si alguna vez ha desempolvado— que diferentes tipos de cosas necesitan diferentes tipos de desempolvado: las chucherías pequeñas se pueden desempolvar con un plumero; las mesas para servir café y las de los extremos podrían necesitar Pledge® y un paño limpio y seco, y los ventiladores del techo podrían necesitar la varita y el cepillo de una aspiradora. La clave en este caso es la diferencia entre lo que describimos con un diagrama y lo que escribimos como parte de nuestro caso de uso.

Nota Podría preguntarse qué tiene que ver desempolvar con los casos de uso y el software. La primera respuesta es que se pueden usar los modelos de caso de uso para cosas que no son software, y la segunda parte es que el software se encuentra en un número cada vez mayor de aparatos. Suponga que estábamos definiendo casos de uso para un robot que limpia casas; entonces nuestras reglas para desempolvar podrían ser útiles. Y si se está preguntando cuán probable podría ser el software para robots, entonces considere la aspiradora Roomba®, un pequeño robot que vaga por un cuarto aspirando los desperdicios y, según su material de mercadeo, incluso sabe cuándo recargarse. Alguien tuvo que definir e implementar esas capacidades.

El caso de uso para desempolvar del párrafo anterior constaría de un actor, “Niña”, un conector de asociación y un caso de uso “Desempolvar la sala” (figura 2-9). No hace falta que el propio diagrama de casos de uso describa todas las microtareas necesarias de las que consta “Desempolvar la sala”. Por ejemplo, “Encontrar Pledge y un paño limpio y seco” es una subtarea necesaria, pero en realidad, no es un caso de uso en y por sí misma. Los casos de uso buenos significan tener que hallar buenos actores y el nivel correcto de detalle, sin hacer confusos los diagramas.

Después de que tenemos el diagrama de caso de uso, podemos agregar información de soporte de la documentación del modelo para nuestro caso de uso. Las funciones primarias incluirían desempolvar las áreas clave, y las funciones secundarias incluirían la preparación, como hacerse de la aspiradora y hallar el Pledge. Los escenarios adecua-

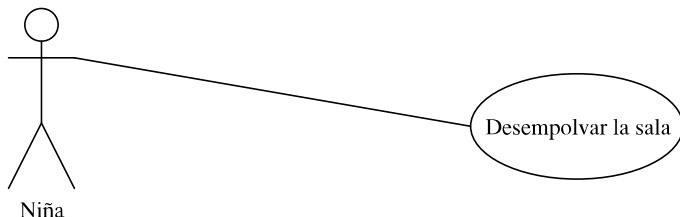


Figura 2-9 Caso de uso para un actor niña y desempolvar una sala.

dos incluirían el manejo de áreas problemas específicas, como desempolvar los marcos de los cuadros y artículos de colección. Los requisitos no funcionales podrían incluir “Terminar de desempolvar antes de que lleguen los abuelos”. No se preocupe acerca de lograr diagramas perfectos ni de la documentación del caso de uso; use el borrador para ayudarse a considerar los detalles y los diagramas de caso de uso para obtener una buena imagen de sus objetivos.

¿Cuántos diagramas son suficientes?

La suficiencia es un problema difícil. Si proporciona demasiados casos de uso, su modelado puede continuar durante meses o incluso años. También puede adentrarse en el mismo problema con la documentación de los casos de uso.

NOTA *Fui consultor en un proyecto para un departamento grande de la agencia de defensa. Literalmente, la agencia había estado trabajando sobre casos de uso por casi dos años sin tener el fin a la vista. Aparte de que me pareció un proyecto interminable, los expertos del dominio sentían que se estaban captando casos de uso erróneos o que las casos de uso tenían poco o ningún valor práctico y explicativo. Los modelos no estaban logrando la marca. El objetivo es captar las características esenciales del objetivo de usted, y los modelos de casos de uso son una excelente manera de baja tecnología para hacer que se involucren expertos en el dominio que no son técnicos. Pasar por alto el valor de incitación al diálogo de los diagramas de casos de uso es perder la mitad del valor de esos diagramas.*

Una línea de base razonable es que las aplicaciones de mediana complejidad podrían tener entre 20 y 50 buenos casos de uso. Si usted sabe que su problema es moderadamente complejo y tiene cinco casos de uso, entonces puede estar pasando por alto funcionalidad crítica. Por otra parte, si tiene cientos de casos de uso, entonces puede estar subdividiendo macrocajas de uso prácticas en microcajas.

Desafortunadamente, no existen reglas difíciles y rápidas. Definir los casos de uso correctos requiere práctica y buen juicio que se adquiere con el transcurso del tiempo. Para ayudarle a empezar a adquirir alguna experiencia, en la siguiente subsección se demuestran algunos diagramas reales de casos de uso para www.motown-jobs.com.

Ejemplos de diagramas de casos de uso

Este libro es acerca del UML. La documentación específica de texto no es parte del UML, de modo que limitaré los ejemplos de esta sección a la creación de los diagramas de casos de uso. Puede usar su imaginación y el borrador de la sección titulada “Documentación de un caso de uso usando un borrador” para practicar la escritura de las descripciones de casos de uso.

Motown-jobs.com es un producto de mi empresa, Software Conceptions, Inc. Motown-jobs es un sitio web para poner en contacto personas que buscan trabajo con quienes los ofrecen; es un sitio web como dice.com, monster.com, computerjobs.com o hotjobs.com y está implementado en ASP.NET. Dejando todo esto aparte, Motown-jobs.com se inició como una idea cuyas características se captaron como un grupo de casos de uso. Debido a que estaba estructurando el software para mi empresa, tuve que representar el papel de experto en el dominio, siendo el dominio lo que se requiere para hacer coincidir patrones con empleados. Dado que he estado buscando y hallando clientes para mi empresa durante 15 años, tengo algo de experiencia en esta área.

Hallar los casos de uso puede iniciar con una entrevista con su experto en el dominio, o bien haciendo una lista. Como yo estaba representando el papel de entrevistador y entrevistado, sencillamente empecé con una lista de las cosas que pensaba que Motown-jobs.com necesitaría ofrecer para ser útil. He aquí mi lista:

- Los patrones o los agentes de los patrones querrán publicar información acerca de los trabajos que están ofreciendo.
- Quienes están buscando trabajo pueden querer publicar un currículum vítae que puedan ver los patrones potenciales.
- Los patrones o los agentes de los patrones querrán buscar en forma activa en el sitio web los currículum vítae que se ajusten a las habilidades necesarias para llenar los sitios vacantes en el trabajo.
- Quienes están buscando empleo querrán buscar en los puestos que se encuentran en lista.
- Los patrones o los agentes de los patrones deberán pagar por las listas y por buscar en los currículum vítae, pero publicar currículum vítae o buscar en las listas de trabajos será un servicio gratuito.
- Una fuente adicional de ingresos podría ser publicidad y servicios de estructuración de currículum vítae, de modo que el sitio web podrá vender y tener espacio para publicidad, y ayudar a los solicitantes de trabajo a crear su currículum vítae.

Además de que escribir software es caro y de que también lo son el hardware, el software del servidor y las conexiones de alta velocidad de Internet tanto para comprarlos como para darles mantenimiento, ayudar a las empresas a encontrar empleados es un servicio valioso, o, por lo menos, ésa es la premisa que se encuentra detrás de la estructuración de Motown-jobs.com. Resolver acerca de cuánto cobrar por las listas y para atraer anunciantes son funciones de negocios y de mercadeo, de modo que hablaré acerca de eso en mi lista de casos de uso.

Ahora bien, claro que podría empeñarme en examinar todas las pequeñas tareas de las que consta cada una de las macrotareas —como publicar las vacantes de puestos de trabajo—, pero la lista que tengo es un buen lugar para iniciar. Empecemos por diagramar estas características (figura 2-10).

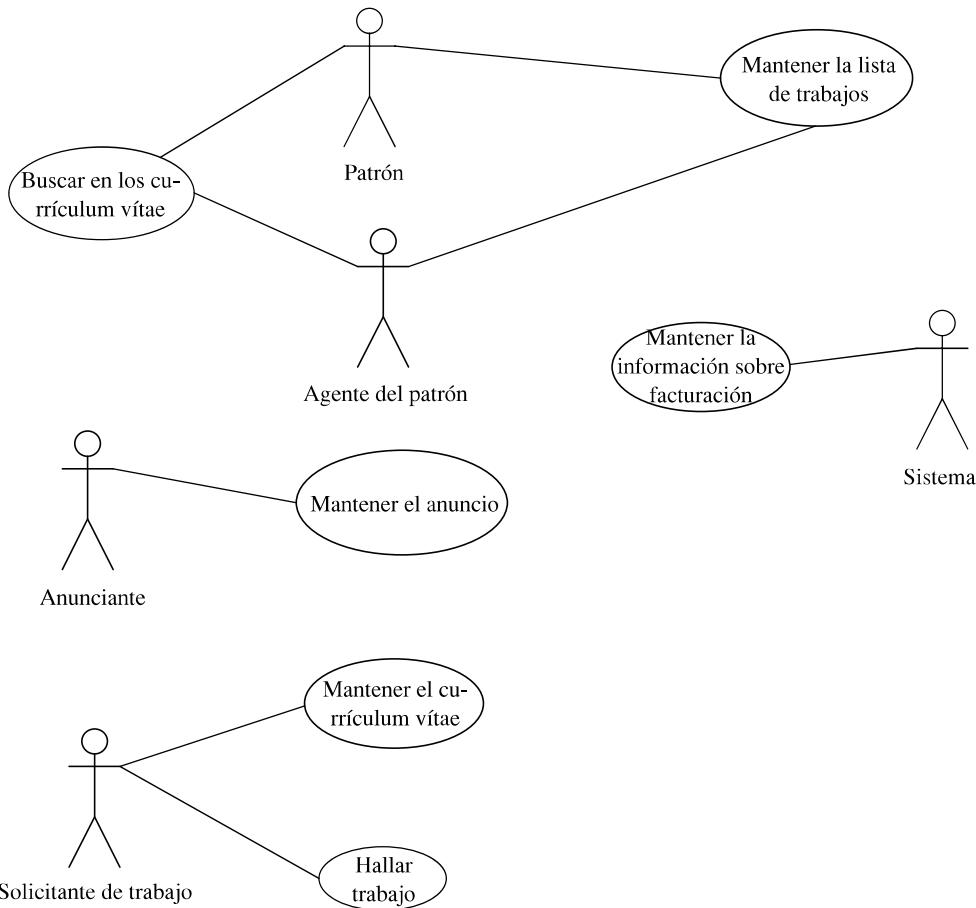


Figura 2-10 Un primer paso en el diagrama de casos de uso para Motown-jobs.com.

Observe en la figura 2-10 que capté mantener trabajos y hallar currículo para la clasificación patrón, mantener anuncios para la clasificación anunciantes, publicar currículo y hallar trabajos para la clasificación solicitantes de empleo y para administrar la facturación para el sistema. Lo siguiente que puedo hacer es preguntar a las partes involucradas si estos casos de uso captan la esencia de las características que necesito.

Como un diagrama de caso de uso, a esto le doy una calificación de C, pero es un inicio. Lo siguiente que puedo hacer es revisar a los actores y a los propios casos de uso en busca de redundancias, simplificaciones o detalles adicionales que se necesitan, y hacer al diagrama los ajustes necesarios.

Definición de los actores

En el diagrama de casos de uso de la figura 2-10, tengo los actores “Patrón” y “Agente del patrón”; no obstante, para todas las intenciones y finalidades, estos dos actores hacen

las mismas cosas en relación con el sistema y lo hacen de la misma manera; por consiguiente, puedo eliminar el “Agente del patrón” y renombrar al “Patrón” como “Propietario de la tarea”; con una descripción sencilla, “Propietario de la tarea”, se capta la idea de que un trabajo en lista “lo posee” una parte responsable. En la figura 2-11, se muestra la revisión en el diagrama de casos de uso.

A continuación, parece bastante obvio que una lista de puestos de trabajo, un currículum vítae y un anuncio son todos clasificaciones de listas, y las personas a quienes pertenecen esos elementos son “Propietarios de listas”. Puede experimentar con estas relaciones usando la generalización. En la figura 2-12, se muestra el diagrama modificado de casos de uso.

En la figura 2-12, se tratan los trabajos, los anuncios y los currículum vítae todos como listas que necesitan mantenerse. También se muestra que el sistema de facturación está asociado con las listas y las búsquedas de los currículum vítae. En ciertos aspectos, la figura 2-12 es una mejora, pero en otros es demasiado ingeniosa. Por ejemplo, describir

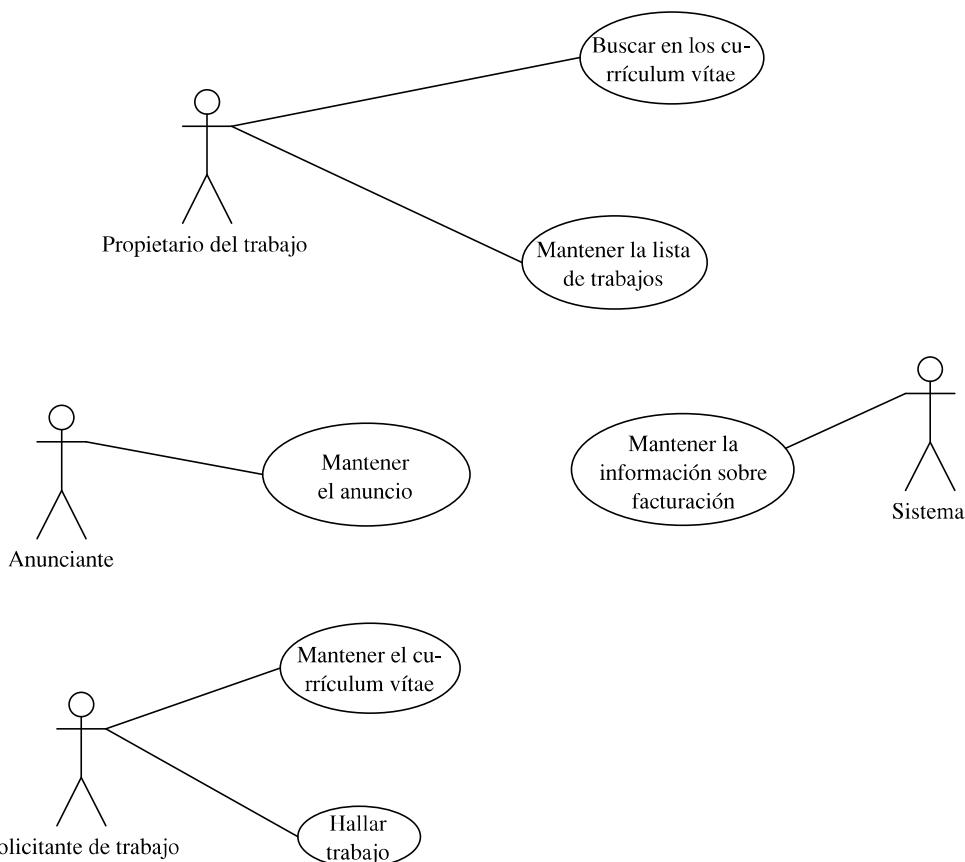


Figura 2-11 “Patrón” y “Agente del patrón” se convierten en un solo actor: “Propietario del trabajo”.

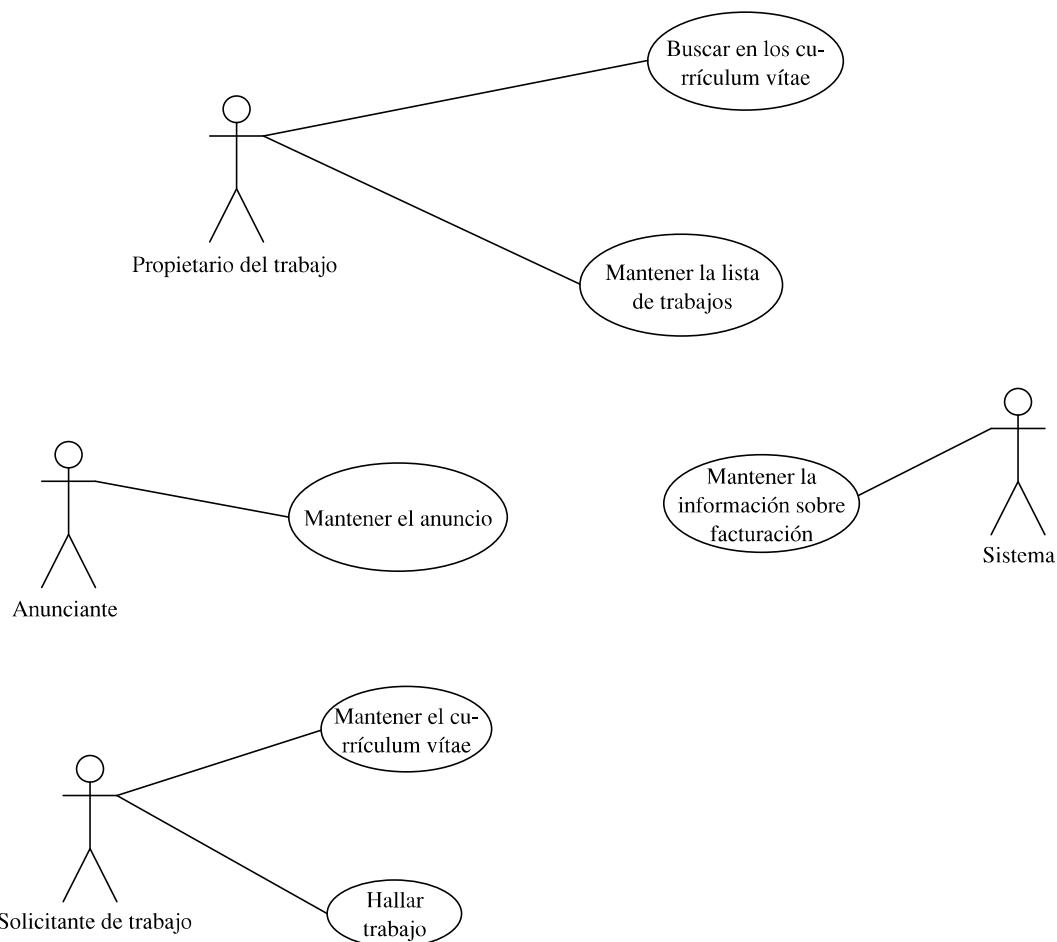


Figura 2-12 Esta figura sugiere que los trabajos, los currículum vitae y los anuncios son todos listas que debe mantener un propietario de lista, así como una asociación entre el sistema de facturación y las listas y las búsquedas de los currículum vitae.

a un solicitante de trabajo como un “Propietario de listas” sugiere que cada solicitante de trabajo posee un currículum vitae en lista. ¿Qué sucede si un solicitante de trabajo no quiere publicar un currículum vitae? Además, dije que publicar un currículum vitae es un servicio gratuito, pero la implicación es que el sistema de facturación trate las listas de currículum vitae como un concepto susceptible de facturación. ¿Significa esto que es susceptible de facturación, pero que el costo es 0 dólares? El diagrama revisado parece un poco más ingenioso y lleva tanto a preguntas como a respuestas. Quizás podría, además, dividir “Listas” en “Listas susceptibles de facturación” y “Listas gratuitas”. Esto podría resolver la cuestión del sistema de facturación, pero ¿qué sucede acerca de los solicitantes de trabajo que no publican currículum vitae? Todavía debo resolver este problema. Por ahora, regreso a los cuatro actores separados, en oposición a los tres tipos de propietarios de listas y el actor sistema (figura 2-13).

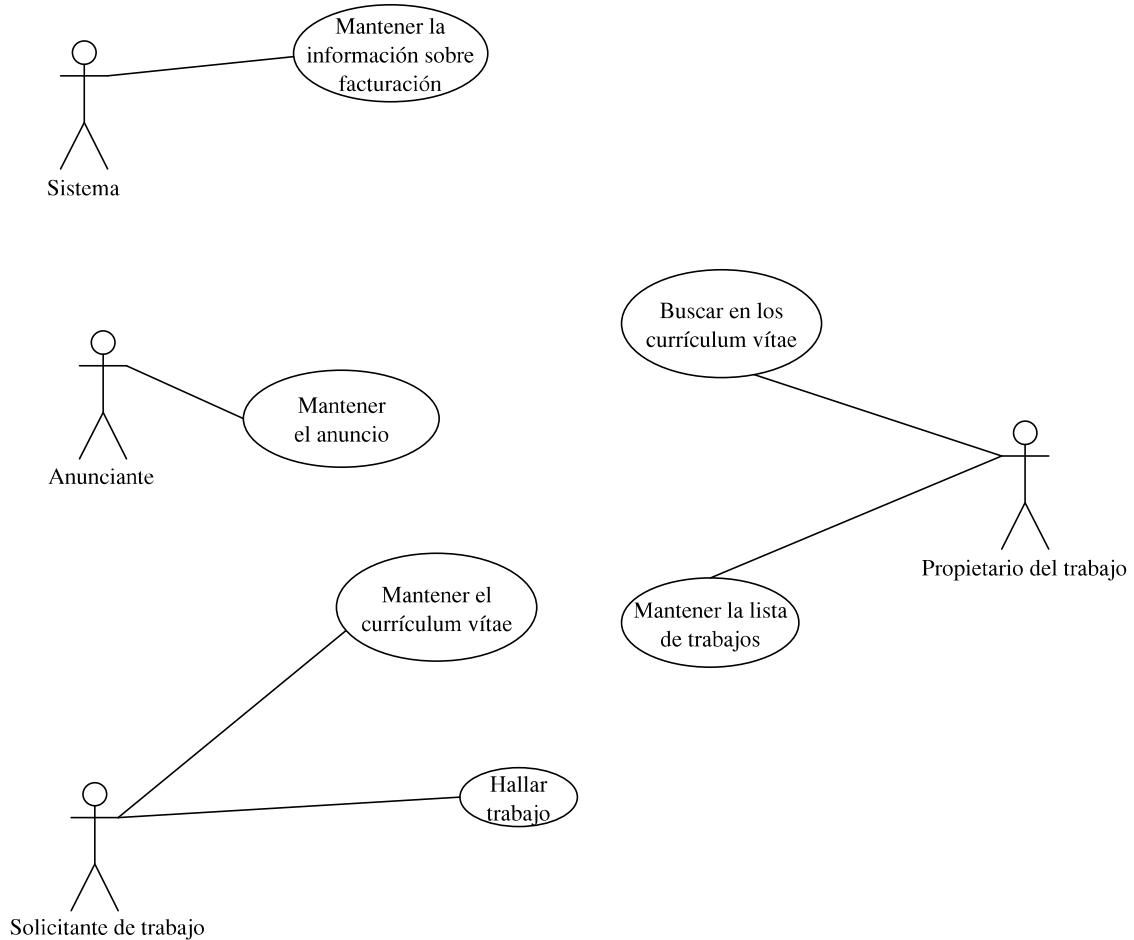


Figura 2-13 Cuatro actores separados no relacionados que participan en casos de uso no relacionados.

Me gusta la forma más sencilla del diagrama de casos de uso de la figura 2-13; está menos abarrotada, es más fácil de seguir y me dice lo que necesito saber acerca de las características del sistema.

División de los casos de uso en diagramas múltiples

Puede elegir tener un diagrama maestro de casos de uso y varios diagramas menores de casos de uso o sólo varios diagramas menores. Usted decide. Los diagramas más sencillos son más fáciles de manejar y seguir, pero puede ser que no muestren cómo están relacionados los casos de uso. En general, prefiero los diagramas sencillos y separados y crear un solo diagrama maestro, si estoy seguro de que al hacerlo obtendré algunos beneficios específicos.

En mi ejemplo de Motown-jobs.com, tengo cuatro facetas significativas; tengo casos de uso relacionados con el solicitante de empleo, casos de uso relacionados con el propietario del trabajo, casos de uso para los anunciantes y el sistema de facturación. Para examinar cada una de estas facetas del sistema, separaré estos casos de uso y los actores que les incumben en diagramas separados y agregaré detalles. En las figuras 2-14 a la 2-17 se muestran los nuevos diagramas.

Al separar “Mantener la información sobre facturación” en un caso de uso separado, tengo espacio para agregar detalles. Por ejemplo, es razonable que el sistema de facturación se interese sólo en lo que es susceptible de facturación y que un actor llamado “Usuario registrado” pueda mantener elementos susceptibles de facturación. Advierta que agregué el caso de uso “Entrar”. Dado que necesito saber cuáles usuarios están para ser facturados, necesitaré un medio de registrar y autenticar.

En la figura 2-15, introduce la idea de que un solicitante de empleo también se considera como usuario registrado. Sin embargo, elijo requerir registro sólo si el usuario quiere publicar un currículum. Quiero saber cuáles personas están proporcionando información a nuestro sistema, pero no lo requiero de los navegadores casuales. Una vez más, para publicar algo en el sistema, requeriré que al usuario se le dé acceso y, de lo contrario, sólo ofrecer al usuario casual la oportunidad de registrarse. El concepto de usuario registrado sugiere que necesito otro caso de uso “Mantener información de registros”. Esto puede implementarse como un sencillo diagrama de casos de uso, con el actor “Usuario registrado” y una asociación al nuevo caso de uso.

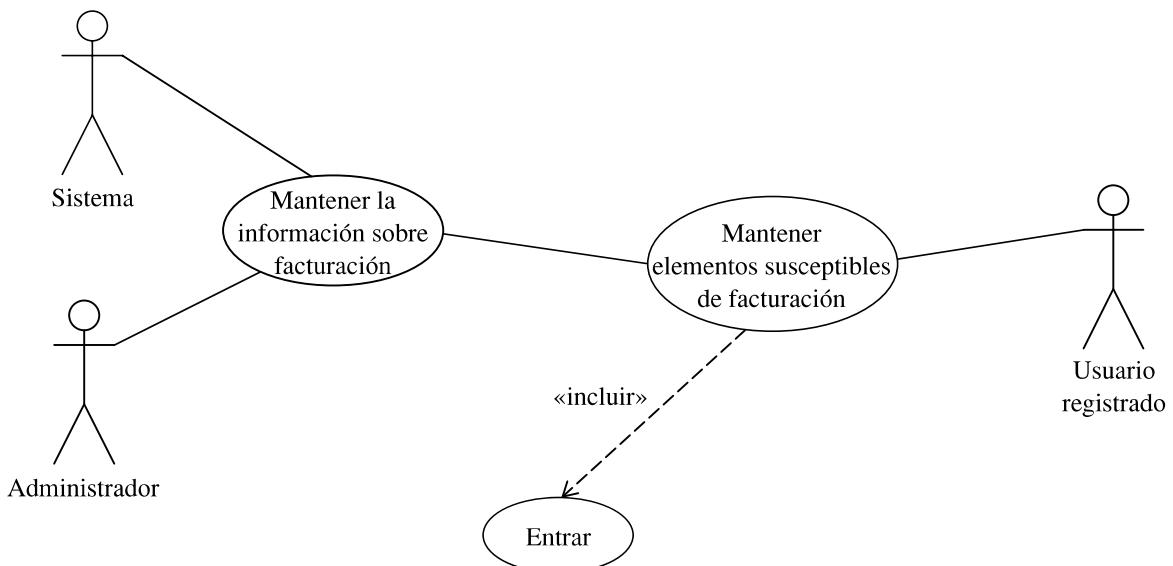


Figura 2-14 En esta figura se muestra que un nuevo actor, llamado “Usuario registrado”, puede mantener un elemento susceptible de facturación, si ese usuario entra y el sistema de facturación se asocia con los elementos susceptibles de facturación.

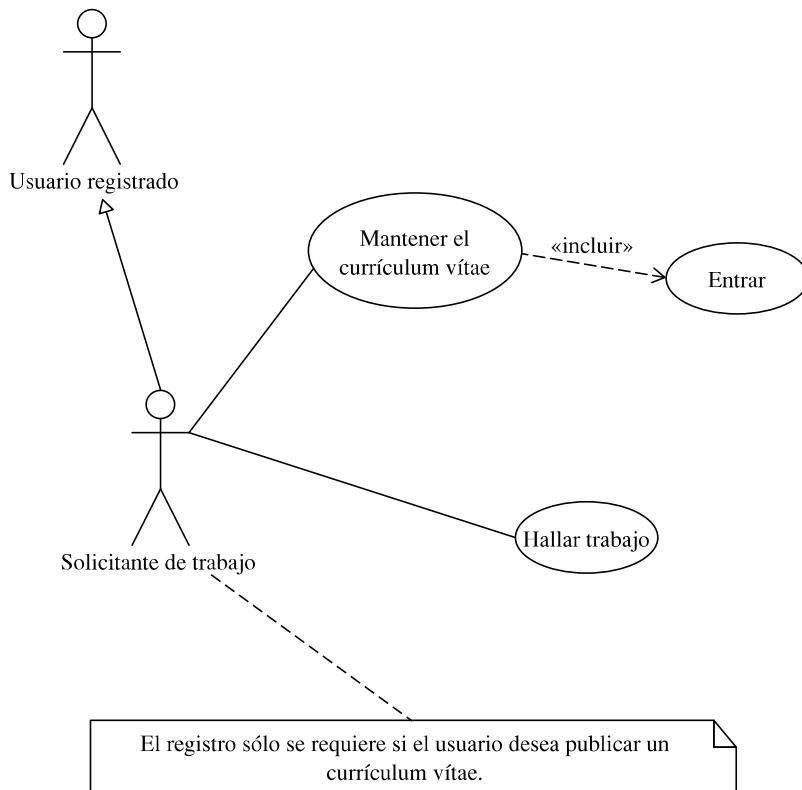


Figura 2-15 Vista ampliada de casos de uso relacionados con los solicitantes de trabajo.

En la figura 2-16, muestro que un anunciante es un usuario registrado y también incluyo que “Mantener el anuncio” generaliza “Mantener elementos susceptibles de facturación”. Dado que “Mantener elementos susceptibles de facturación” también está en el diagrama de la figura 2-16, de igual manera sé que esto significa que estoy ligado con los casos de uso de facturación, registro y autenticación (o concesión de acceso), pero de manera intencional quité esos elementos del diagrama para no abarrotarlo.

En la figura 2-17, señalo la dependencia entre “Mantener elementos susceptibles de facturación” y “Entrar” mostrando el conector de dependencia entre estos dos casos de uso. Debe resultar obvio que, como “Buscar en los currículum vitae” y “Mantener la lista de trabajos” generalizan “Mantener elementos susceptibles de facturación”, se requiere la autenticación para publicar trabajos y buscar en los currículum vitae. El uso de un solo conector simplifica el diagrama.

Ciertamente, será bienvenido el que usted intente crear un solo diagrama maestro de casos de uso, pero no necesita hacerlo. Incluso en este sistema relativamente sencillo, un solo modelo monolítico podría únicamente agregarse a la confusión; nuestro objetivo es reducir la confusión y aumentar la comprensión tan sencilla y directamente como sea posible.

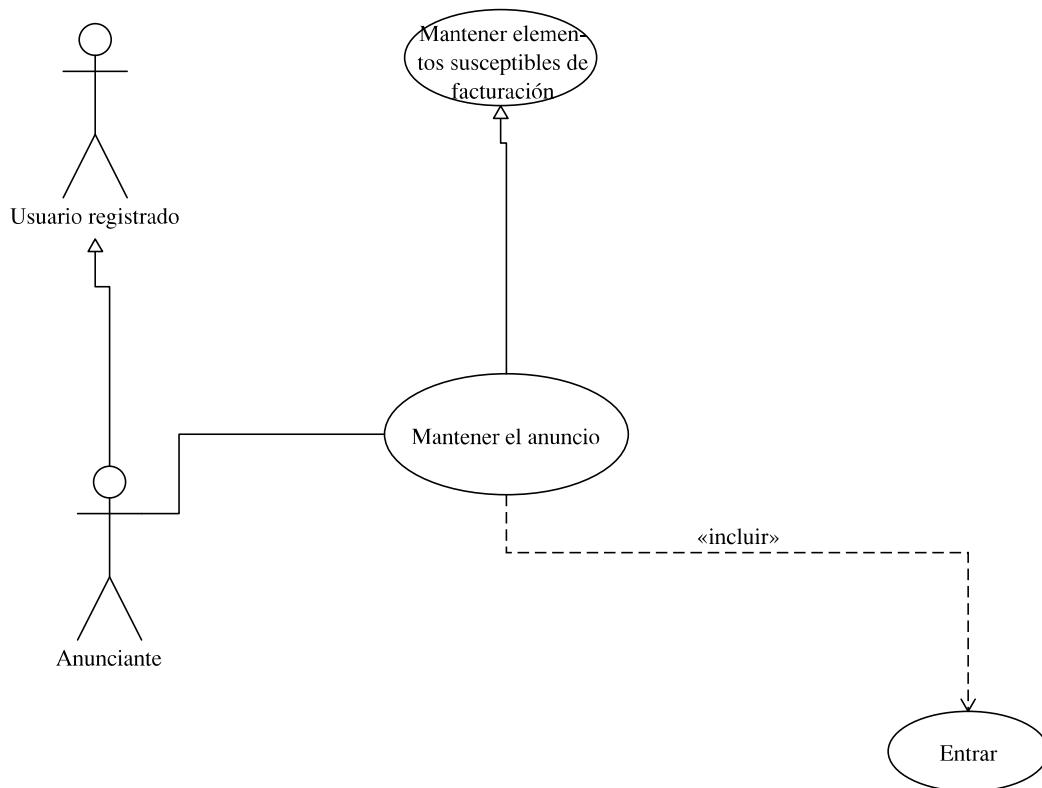


Figura 2-16 Vista cada vez más detallada de los casos de uso en los que intervienen los anunciantes.

Pienso que estos cuatro modelos hacen esto, pero la exposición ilustra en forma precisa la clasificación de temas que deberá pesar al decidir en cuáles modelos invertir su tiempo.

Manera de hallar la línea final

A medida que se evalúen sus diagramas de casos de uso y su documentación como texto escrito, le surgirán otras ideas y las cosas que pasó por alto. Esto es de esperarse. Documente estas ideas, incluso si al final las descarta. También esté preparado para revisar sus modelos a medida que cambien su comprensión y la de sus clientes o el clima de la empresa. Una comprensión creciente o un clima dinámico de la empresa significa más diagramas de casos de uso y revisiones a los ya existentes. Si anticipa la naturaleza dinámica de la comprensión, entonces no tendrá problema para continuar con los pasos siguientes, en lugar de intentar crear un juego perfecto de casos de uso sin reflexionar.

El objetivo de crear diagramas de casos de uso es documentar los aspectos importantes del sistema, para proporcionar a los usuarios una manera de baja tecnología para evaluar en forma visual sus comprensiones mutuas y, a continuación, seguir adelante. El resultado que deseamos es un juego de casos de uso “suficientemente bueno”, no perfecto.

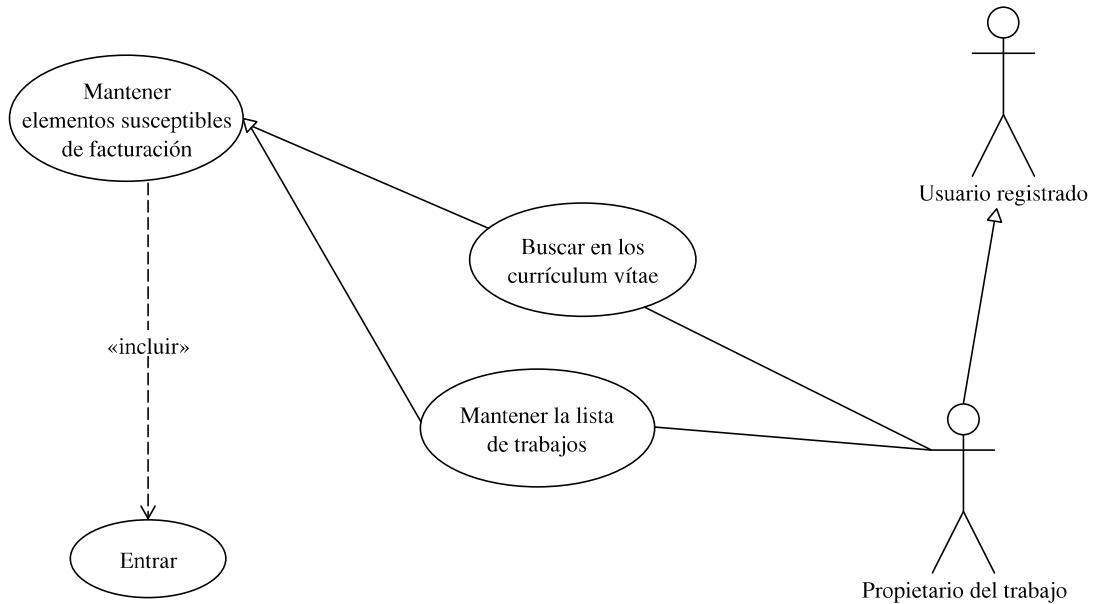


Figura 2-17 En esta figura se muestra la relación entre el propietario del trabajo y sus casos de uso, incluyendo una descripción clara de que se requiere la autenticación y que ese propietario esté administrando elementos susceptibles de facturación.

Diseño controlado con casos de uso

Hasta ahora, he definido casos de uso significativos y los diagramas de casos de uso para Motown-jobs.com. (Dejé fuera “Mantener información de los clientes”, pero sé que la necesito.) Con base en la exposición, debe resultar obvio que omití tareas menores, por ejemplo, leer listas en una base de datos y escribirlas para ésta. Sin embargo, esto queda cubierto en “Mantener la lista de trabajos”. No necesito un diagrama separado de casos de uso para mostrar que estoy “clabeando” —por CLAB, o sea, crear, leer, actualizar y borrar— listas, anuncios o currículum vitae, aunque resultará útil describir estas cosas en diagramas futuros, como los diagramas de secuencia (vea el capítulo 6 para obtener más información). Lo siguiente que me interesa realizar es el establecimiento de prioridades.

Demasiados proyectos pasan por alto por completo los casos de uso e ignoran el establecimiento de prioridades, pero los casos de uso existen para ayudarle a administrar el alcance y para establecer prioridades. El término *diseño controlado por casos de uso* significa que expresamos lo que estamos estructurando en nuestros casos de uso con el fin de limitar el alcance y evitar el desperdicio de tiempo, y establecemos prioridades en lo que estructuramos empezando con las características más críticas y de prioridad más alta. Con demasiada frecuencia, los programadores estructurarán cosas agradables o fáciles como primeros diálogos “Acerca de” y campanas y silbidos innecesarios, porque están examinando alguna nueva tecnología, y esto es un factor significativo de por qué fallan tantos proyectos.

Después de que haya definido sus casos de uso, querrá establecer prioridades y además diseñar e implementar una solución para apoyar aquellos casos de uso con la prioridad más alta o que representan el riesgo más significativo. ¿Cómo decide qué diseñar y estructurar primero? La respuesta es pregúntele a su cliente qué es lo más riesgoso, lo más importante o lo más valioso y a continuación enfoque sus energías en esos casos de uso.

Nota *La pregunta real que debe hacer a su cliente es: “¿Qué características podemos estructurar primero de modo que si estamos fuera de tiempo y de presupuesto, todavía tendremos un producto que pueda comercializarse?” Los clientes no siempre quieren escuchar las preguntas difíciles, y usted deberá aplicar cierta diplomacia, pero hallar la respuesta correcta a esta pregunta y actuar en términos de ella puede ser lo más importante que usted haga.*

Para Motown-jobs.com, decidí —como cliente— que puedo presentarme en el mercado con un servicio de listas de trabajo con base en una tarifa. Esto significa que si implemento “Mantener la lista de trabajos”, “Buscar un trabajo” y “Mantener la información sobre facturación”, tendré un producto con el que puedo entrar al mercado. Esto no significa que no querré estructurar en el sistema la publicación de currículum vítae, la búsqueda y el apoyo para la publicidad; sólo significa que éstas no son las características más importantes.

Las prioridades que siguen son más difíciles. ¿Debo estructurar a continuación la publicación de currículum vítae y la búsqueda, o la publicidad? La respuesta es que quiero que los solicitantes de trabajo usen el servicio y los propietarios del trabajo vean que hay una gran cantidad de tráfico e interés en mi sitio, de modo que apoyaré a continuación la publicación de un currículum vítae —el cual es un servicio gratuito pero crítico— y, después, la búsqueda de currículum vítae, el cual también es un servicio gratuito, pero que depende de tener currículum vítae para revisar. Por último, apoyaré la publicidad, la cual finalmente depende de tener tráfico suficiente para interesar a los anunciantes.

Lo importante aquí es que la identificación de mis casos de uso me ayudó a establecer prioridades en mi lista de tareas e ilustra un camino crítico para mi criterio de éxito mínimo: vender anuncios de se solicitan empleados.

Examen

1. ¿Qué símbolo representa un caso de uso?
 - a. Una línea
 - b. Una línea dirigida
 - c. Una figura de palillos
 - d. Un óvalo que contiene texto

2. Un actor solamente puede serlo una persona.
 - a. Verdadero
 - b. Falso
3. ¿Qué símbolo representa una dependencia?
 - a. Una línea
 - b. Una línea con un triángulo que apunta hacia el elemento dependiente
 - c. Una línea punteada con una flecha que apunta hacia el elemento dependiente
 - d. Una línea punteada con una flecha que apunta hacia el elemento del que se depende
4. ¿Cómo se indica un estereotipo sobre un conector?
 - a. Texto entre un par de comillas angulares
 - b. Texto llano próximo al conector
 - c. La palabra estereotipo dentro del símbolo de óvalo
5. Se usa una relación de inclusión para reutilizar el comportamiento modelado por otro caso de uso.
 - a. Verdadero
 - b. Falso
6. Se usa una relación de extensión para modelar características opcionales del sistema.
 - a. Verdadero
 - b. Falso
7. En el UML la generalización se refleja en la implementación por
 - a. polimorfismo.
 - b. agregación.
 - c. herencia.
 - d. interfaces.
8. Todas las capacidades de un sistema deben representarse por un caso de uso.
 - a. Verdadero
 - b. Falso
9. En una relación extendida, la flecha apunta hacia el
 - a. caso de uso básico.
 - b. caso de uso de extensión.