CAPÍTULO

5

COMPRENSIÓN DE LOS REQUERIMIENTOS

CONCEPTOS CLAVE

Entender los requerimientos de un problema es una de las tareas más difíciles que enfrenta el ingeniero de software. Cuando se piensa por primera vez, no parece tan difícil desarrollar un entendimiento claro de los requerimientos. Después de todo, ¿acaso no sabe el cliente lo que se necesita? ¿No deberían tener los usuarios finales una buena comprensión de las características y funciones que le darán un beneficio? Sorprendentemente, en muchas instancias la respuesta a estas preguntas es "no". E incluso si los clientes y los usuarios finales explican sus necesidades, éstas cambiarán mientras se desarrolla el proyecto.

En el prólogo a un libro escrito por Ralph Young [You01] sobre las prácticas eficaces respecto de los requerimientos, escribí lo siguiente:

> Es la peor de las pesadillas. Un cliente entra a la oficina, toma asiento, lo mira a uno fijamente a los ojos y dice: "Sé que cree que entiende lo que digo, pero lo que usted no entiende es que lo que digo no es lo que quiero decir." Invariablemente, esto pasa cuando ya está avanzado el proyecto, después de que se han hecho compromisos con los plazos de entrega, que hay reputaciones en juego y mucho dinero invertido.
>
> Todos los que hemos trabajado en el negocio de los sistemas y del software durante algunos años hemos vivido la pesadilla descrita, pero pocos hemos aprendido a escapar. Batallamos cuando tratamos de obtener los requerimientos de nuestros clientes. Tenemos problemas para entender la información que obtenemos. Es frecuente que registremos los requerimientos de manera desorganizada y que dediquemos muy poco tiempo a verificar lo que registramos. Dejamos que el cambio nos controle en lugar de establecer mecanismos para controlarlo a él. En pocas palabras, fallamos en establecer un fundamento sólido para el sistema o software. Cada uno de los problemas es difícil. Cuando se combinan, el panorama es atemorizador aun para los gerentes y profesionales más experimentados. Pero hay solución.

UNA MIRADA RÁPIDA

¿Qué es? Antes de comenzar cualquier trabajo técnico es una buena idea aplicar un conjunto de tareas de ingeniería a los requerimientos. Éstas llevarán a la comprensión de cuál será el efecto que tendrá el software en el negocio, qué es lo que quiere el cliente y cómo interactuarán los usuarios finales con el software.

¿Quién lo hace? Los ingenieros de software (que en el mundo de las tecnologías de información a veces son llamados *ingenieros de sistemas* o *analistas*) y todos los demás participantes del proyecto (gerentes, clientes y usuarios) intervienen en la ingeniería de requerimientos.

¿Por qué es importante? Diseñar y construir un elegante programa de cómputo que resuelva el problema equivocado no satisface las necesidades de nadie. Por eso es importante entender lo que el cliente desea antes de comenzar a diseñar y a construir un sistema basado en computadora.

¿Cuáles son los pasos? La ingeniería de requerimientos comienza con la concepción, tarea que define el alcance y la naturaleza del problema que se va a resolver. Va seguida de la indagación, labor que ayuda a los participantes a definir lo que se requiere. Después sigue la elaboración, donde se refinan y modifican los requerimientos básicos. Cuando los participantes definen el problema, tiene lugar una negociación: ¿cuáles son las prioridades, qué es lo esencial, cuándo se requiere? Por último, se especifica el problema de algún modo y luego se revisa o valida para garantizar que hay coincidencia entre la comprensión que usted tiene del problema y la que tienen los participantes.

¿Cuál es el producto final? El objetivo de los requerimientos de ingeniería es proporcionar a todas las partes un entendimiento escrito del problema. Esto se logra por medio de varios productos del trabajo: escenarios de uso, listas de funciones y de características, modelos de requerimientos o especificaciones.

¿Cómo me aseguro de que lo hice bien? Se revisan con los participantes los productos del trabajo de la ingeniería de requerimientos a fin de asegurar que lo que se aprendió es lo que ellos quieren decir en realidad. Aquí cabe una advertencia: las cosas cambiarán aun después de que todas las partes estén de acuerdo, y seguirán cambiando durante todo el proyecto.

Es razonable afirmar que las técnicas que se estudiarán en este capítulo no son una "solución" verdadera para los retos que se mencionaron, pero sí proveen de un enfoque sólido para enfrentarlos.

5.1 Ingeniería de requerimientos

Cita:

"La parte más difícil al construir un sistema de software es decidir qué construir. Ninguna parte del trabajo invalida tanto al sistema resultante si ésta se hace mal. Nada es más difícil de corregir después."

Fred Brooks

El diseño y construcción de software de computadora es difícil, creativo y sencillamente divertido. En realidad, elaborar software es tan atractivo que muchos desarrolladores de software quieren ir directo a él antes de haber tenido el entendimiento claro de lo que se necesita. Argumentan que las cosas se aclararán a medida que lo elaboren, que los participantes en el proyecto podrán comprender sus necesidades sólo después de estudiar las primeras iteraciones del software, que las cosas cambian tan rápido que cualquier intento de entender los requerimientos en detalle es una pérdida de tiempo, que las utilidades salen de la producción de un programa que funcione y que todo lo demás es secundario. Lo que hace que estos argumentos sean tan seductores es que tienen algunos elementos de verdad.[1] Pero todos son erróneos y pueden llevar un proyecto de software al fracaso.

El espectro amplio de tareas y técnicas que llevan a entender los requerimientos se denomina *ingeniería de requerimientos*. Desde la perspectiva del proceso del software, la ingeniería de requerimientos es una de las acciones importantes de la ingeniería de software que comienza durante la actividad de comunicación y continúa en la de modelado. Debe adaptarse a las necesidades del proceso, del proyecto, del producto y de las personas que hacen el trabajo.

PUNTO CLAVE

La ingeniería de requerimientos establece una base sólida para el diseño y la construcción. Sin ésta, el software resultante tiene alta probabilidad de no satisfacer las necesidades del cliente.

La ingeniería de requerimientos tiende un puente para el diseño y la construcción. Pero, ¿dónde se origina el puente? Podría argumentarse que principia en los pies de los participantes en el proyecto (por ejemplo, gerentes, clientes y usuarios), donde se definen las necesidades del negocio, se describen los escenarios de uso, se delinean las funciones y características y se identifican las restricciones del proyecto. Otros tal vez sugieran que empieza con una definición más amplia del sistema, donde el software no es más que un componente del dominio del sistema mayor. Pero sin importar el punto de arranque, el recorrido por el puente lo lleva a uno muy alto sobre el proyecto, lo que le permite examinar el contexto del trabajo de software que debe realizarse; las necesidades específicas que deben abordar el diseño y la construcción; las prioridades que guían el orden en el que se efectúa el trabajo, y la información, las funciones y los comportamientos que tendrán un profundo efecto en el diseño resultante.

CONSEJO

Espere hacer un poco de diseño al recabar los requerimientos, y un poco de requerimientos durante el trabajo de diseño.

La ingeniería de requerimientos proporciona el mecanismo apropiado para entender lo que desea el cliente, analizar las necesidades, evaluar la factibilidad, negociar una solución razonable, especificar la solución sin ambigüedades, validar la especificación y administrar los requerimientos a medida de que se transforman en un sistema funcional [Tha97]. Incluye siete tareas diferentes: concepción, indagación, elaboración, negociación, especificación, validación y administración. Es importante notar que algunas de estas tareas ocurren en paralelo y que todas se adaptan a las necesidades del proyecto.

Cita:

"Las semillas de los desastres enormes del software por lo general se vislumbran en los tres primeros meses del inicio del proyecto."

Coper Jones

Concepción. ¿Cómo inicia un proyecto de software? ¿Existe un solo evento que se convierte en el catalizador de un nuevo sistema o producto basado en computadora o la necesidad evoluciona en el tiempo? No hay respuestas definitivas a estas preguntas. En ciertos casos, una conversación casual es todo lo que se necesita para desencadenar un trabajo grande de ingeniería de software. Pero en general, la mayor parte de proyectos comienzan cuando se identifica una necesidad del negocio o se descubre un nuevo mercado o servicio potencial. Los partici-

1 Esto es cierto en particular para los proyectos pequeños (menos de un mes) y muy pequeños, que requieren relativamente poco esfuerzo de software sencillo. A medida que el software crece en tamaño y complejidad, estos argumentos comienzan a ser falsos.

pantes de la comunidad del negocio (por ejemplo, los directivos, personal de mercadotecnia, gerentes de producto, etc.) definen un caso de negocios para la idea, tratan de identificar el ritmo y profundidad del mercado, hacen un análisis de gran visión de la factibilidad e identifican una descripción funcional del alcance del proyecto. Toda esta información está sujeta a cambio, pero es suficiente para desencadenar análisis con la organización de ingeniería de software.[2]

En la concepción del proyecto,[3] se establece el entendimiento básico del problema, las personas que quieren una solución, la naturaleza de la solución que se desea, así como la eficacia de la comunicación y colaboración preliminares entre los otros participantes y el equipo de software.

Indagación. En verdad que parece muy simple: preguntar al cliente, a los usuarios y a otras personas cuáles son los objetivos para el sistema o producto, qué es lo que va a lograrse, cómo se ajusta el sistema o producto a las necesidades del negocio y, finalmente, cómo va a usarse el sistema o producto en las operaciones cotidianas. Pero no es simple: es muy difícil.

Christel y Kang [Cri92] identificaron cierto número de problemas que se encuentran cuando ocurre la indagación:

¿Por qué es difícil llegar al entendimiento claro de lo que quiere el cliente?

- **Problemas de alcance.** La frontera de los sistemas está mal definida o los clientes o usuarios finales especifican detalles técnicos innecesarios que confunden, más que clarifican, los objetivos generales del sistema.
- **Problemas de entendimiento.** Los clientes o usuarios no están completamente seguros de lo que se necesita, comprenden mal las capacidades y limitaciones de su ambiente de computación, no entienden todo el dominio del problema, tienen problemas para comunicar las necesidades al ingeniero de sistemas, omiten información que creen que es "obvia", especifican requerimientos que están en conflicto con las necesidades de otros clientes o usuarios, o solicitan requerimientos ambiguos o que no pueden someterse a prueba.
- **Problemas de volatilidad.** Los requerimientos cambian con el tiempo.

Para superar estos problemas, debe enfocarse la obtención de requerimientos en forma organizada.

Elaboración. La información obtenida del cliente durante la concepción e indagación se expande y refina durante la elaboración. Esta tarea se centra en desarrollar un modelo refinado de los requerimientos (véanse los capítulos 6 y 7) que identifique distintos aspectos de la función del software, su comportamiento e información.

CONSEJO

La elaboración es algo bueno, pero hay que saber cuándo detenerse. La clave es describir el problema en forma que establezca una base firme para el diseño. Si se trabaja más allá de este punto, se está haciendo diseño.

La elaboración está motivada por la creación y mejora de escenarios de usuario que describan cómo interactuará el usuario final (y otros actores) con el sistema. Cada escenario de usuario se enuncia con sintaxis apropiada para extraer clases de análisis, que son entidades del dominio del negocio visibles para el usuario final. Se definen los atributos de cada clase de análisis y se identifican los servicios[4] que requiere cada una de ellas. Se identifican las relaciones y colaboración entre clases, y se producen varios diagramas adicionales.

Negociación. No es raro que los clientes y usuarios pidan más de lo que puede lograrse dado lo limitado de los recursos del negocio. También es relativamente común que distintos clientes

---

2 Si va a desarrollarse un sistema basado en computadora, los análisis comienzan en el contexto de un proceso de ingeniería de sistemas. Para más detalles de la ingeniería de sistemas, visite el sitio web de esta obra.

3 Recuerde que el proceso unificado (véase el capítulo 2) define una "fase de concepción" más amplia que incluye las fases de concepción, indagación y elaboración, que son estudiadas en dicho capítulo.

4 Un *servicio* manipula los datos agrupados por clase. También se utilizan los términos *operación* y *método*. Si no está familiarizado con conceptos de la orientación a objetos, consulte el apéndice 2, en el que se presenta una introducción básica.

En una negociación eficaz no debe haber ganador ni perdedor. Ambos lados ganan porque un "trato" con el que ambas partes pueden vivir es algo sólido.

o usuarios propongan requerimientos conflictivos con el argumento de que su versión es "esencial para nuestras necesidades especiales".

Estos conflictos deben reconciliarse por medio de un proceso de negociación. Se pide a clientes, usuarios y otros participantes que ordenen sus requerimientos según su prioridad y que después analicen los conflictos. Con el empleo de un enfoque iterativo que da prioridad a los requerimientos, se evalúa su costo y riesgo, y se enfrentan los conflictos internos; algunos requerimientos se eliminan, se combinan o se modifican de modo que cada parte logre cierto grado de satisfacción.

Especificación. En el contexto de los sistemas basados en computadora (y software), el término *especificación* tiene diferentes significados para distintas personas. Una especificación puede ser un documento escrito, un conjunto de modelos gráficos, un modelo matemático formal, un conjunto de escenarios de uso, un prototipo o cualquier combinación de éstos.

La formalidad y el formato de una especificación varían con el tamaño y complejidad del software que se va a construir.

Algunos sugieren que para una especificación debe desarrollarse y utilizarse una "plantilla estándar" [Som97], con el argumento de que esto conduce a requerimientos presentados en forma consistente y por ello más comprensible. Sin embargo, en ocasiones es necesario ser flexible cuando se desarrolla una especificación. Para sistemas grandes, el mejor enfoque puede ser un documento escrito que combine descripciones en un lenguaje natural con modelos gráficos. No obstante, para productos o sistemas pequeños que residan en ambientes bien entendidos, quizá todo lo que se requiera sea escenarios de uso.

INFORMACIÓN

Formato de especificación de requerimientos de software

Una *especificación de requerimientos de software* (ERS) es un documento que se crea cuando debe especificarse una descripción detallada de todos los aspectos del software que se va a elaborar, antes de que el proyecto comience. Es importante notar que una ERS formal no siempre está en forma escrita. En realidad, hay muchas circunstancias en las que el esfuerzo dedicado a la ERS estaría mejor aprovechado en otras actividades de la ingeniería de software. Sin embargo, se justifica la ERS cuando el software va a ser desarrollado por una tercera parte, cuando la falta de una especificación crearía problemas severos al negocio, si un sistema es complejo en extremo o si se trata de un negocio de importancia crítica.

Karl Wiegers [Wie03], de la empresa Process Impact Inc., desarrolló un formato útil (disponible en **www.processimpact.com/process_assets/srs_template.doc**) que sirve como guía para aquellos que deben crear una ERS completa. Su contenido normal es el siguiente:

Tabla de contenido
Revisión de la historia

1. Introducción
1.1 Propósito
1.2 Convenciones del documento
1.3 Audiencia objetivo y sugerencias de lectura
1.4 Alcance del proyecto
1.5 Referencias

2. Descripción general
2.1 Perspectiva del producto
2.2 Características del producto
2.3 Clases y características del usuario
2.4 Ambiente de operación
2.5 Restricciones de diseño e implementación
2.6 Documentación para el usuario
2.7 Suposiciones y dependencias

3. Características del sistema
3.1 Característica 1 del sistema
3.2 Característica 2 del sistema (y así sucesivamente)

4. Requerimientos de la interfaz externa
4.1 Interfaces de usuario
4.2 Interfaces del hardware
4.3 Interfaces del software
4.4 Interfaces de las comunicaciones

5. Otros requerimientos no funcionales
5.1 Requerimientos de desempeño
5.2 Requerimientos de seguridad
5.3 Requerimientos de estabilidad
5.4 Atributos de calidad del software

6. Otros requerimientos

Apéndice A: Glosario
Apéndice B: Modelos de análisis
Apéndice C: Lista de conceptos

Puede obtenerse una descripción detallada de cada ERS si se descarga el formato desde la URL mencionada antes.

Validación. La calidad de los productos del trabajo que se generan como consecuencia de la ingeniería de los requerimientos se evalúa durante el paso de validación. La validación de los requerimientos analiza la especificación[5] a fin de garantizar que todos ellos han sido enunciados sin ambigüedades; que se detectaron y corrigieron las inconsistencias, las omisiones y los errores, y que los productos del trabajo se presentan conforme a los estándares establecidos para el proceso, el proyecto y el producto.

Un aspecto clave durante la validación de los requerimientos es la consistencia. Utilice el modelo de análisis para asegurar que los requerimientos se han enunciado de manera consistente.

El mecanismo principal de validación de los requerimientos es la revisión técnica (véase el capítulo 15). El equipo de revisión que los valida incluye ingenieros de software, clientes, usuarios y otros participantes, que analizan la especificación en busca de errores de contenido o de interpretación, de aspectos en los que tal vez se requiera hacer aclaraciones, falta de información, inconsistencias (problema notable cuando se hace la ingeniería de productos o sistemas grandes) y requerimientos en conflicto o irreales (no asequibles).

INFORMACIÓN

Lista de verificación para validar requerimientos

Con frecuencia es útil analizar cada requerimiento en comparación con preguntas de verificación. A continuación se presentan algunas:

- ¿Los requerimientos están enunciados con claridad? ¿Podrían interpretarse mal?
- ¿Está identificada la fuente del requerimiento (por ejemplo, una persona, reglamento o documento)? ¿Se ha estudiado el planteamiento final del requerimiento en comparación con la fuente original?
- ¿El requerimiento está acotado en términos cuantitativos?
- ¿Qué otros requerimientos se relacionan con éste? ¿Están comparados con claridad por medio de una matriz de referencia cruzada u otro mecanismo?
- ¿El requerimiento viola algunas restricciones del dominio?
- ¿Puede someterse a prueba el requerimiento? Si es así, ¿es posible especificar las pruebas (en ocasiones se denominan criterios de validación) para ensayar el requerimiento?
- ¿Puede rastrearse el requerimiento hasta cualquier modelo del sistema que se haya creado?
- ¿Es posible seguir el requerimiento hasta los objetivos del sistema o producto?
- ¿La especificación está estructurada en forma que lleva a entenderlo con facilidad, con referencias y traducción fáciles a productos del trabajo más técnicos?
- ¿Se ha creado un índice para la especificación?
- ¿Están enunciadas con claridad las asociaciones de los requerimientos con las características de rendimiento, comportamiento y operación? ¿Cuáles requerimientos parecen ser implícitos?

Administración de los requerimientos. Los requerimientos para sistemas basados en computadora cambian, y el deseo de modificarlos persiste durante toda la vida del sistema. La administración de los requerimientos es el conjunto de actividades que ayudan al equipo del proyecto a identificar, controlar y dar seguimiento a los requerimientos y a sus cambios en cualquier momento del desarrollo del proyecto.[6] Muchas de estas actividades son idénticas a las técnicas de administración de la configuración del software (TAS) que se estudian en el capítulo 22.

---

5 Recuerde que la naturaleza de la especificación variará con cada proyecto. En ciertos casos, la "especificación" no es más que un conjunto de escenarios de usuario. En otros, la especificación tal vez sea un documento que contiene escenarios, modelos y descripciones escritas.

6 La administración formal de los requerimientos sólo se practica para proyectos grandes que tienen cientos de requerimientos identificables. Para proyectos pequeños, esta actividad tiene considerablemente menos formalidad.

HERRAMIENTAS DE SOFTWARE

Ingeniería de requerimientos

Objetivo: Las herramientas de la ingeniería de los requerimientos ayudan a reunir éstos, a modelarlos, administrarlos y validarlos.

Mecánica: La mecánica de las herramientas varía. En general, éstas elaboran varios modelos gráficos (por ejemplo, UML) que ilustran los aspectos de información, función y comportamiento de un sistema. Estos modelos constituyen la base de todas las demás actividades del proceso de software.

Herramientas representativas:[7]

En el sitio de Volere Requirements, en **www.volere.co.uk/tools.htm**, se encuentra una lista razonablemente amplia (y actualizada) de herramientas para la ingeniería de requerimientos. En los capítulos 6 y 7 se estudian las herramientas que sirven para modelar aquéllos. Las que se mencionan a continuación se centran en su administración.

EasyRM, desarrollada por Cybernetic Intelligence GmbH (**www.easy-rm.com**), construye un diccionario/glosario especial para proyectos, que contiene descripciones y atributos detallados de los requerimientos.

Rational RequisitePro, elaborada por Rational Software (**www-306.ibm.com/software/awdtools/reqpro/**), permite a los usuarios construir una base de datos de requerimientos, representar relaciones entre ellos y organizarlos, indicar su prioridad y rastrearlos.

En el sitio de Volere ya mencionado, se encuentran muchas herramientas adicionales para administrar requerimientos, así como en la dirección **www.jiludwig.com/Requirements_Management_Tools.html**

5.2 ESTABLECER LAS BASES

En el caso ideal, los participantes e ingenieros de software trabajan juntos en el mismo equipo.[8] En esas condiciones, la ingeniería de requerimientos tan sólo consiste en sostener conversaciones significativas con colegas que sean miembros bien conocidos del equipo. Pero es frecuente que en la realidad esto sea muy diferente.

Los clientes o usuarios finales tal vez se encuentren en ciudades o países diferentes, quizá sólo tengan una idea vaga de lo que se requiere, puede ser que tengan opiniones en conflicto sobre el sistema que se va a elaborar, que posean un conocimiento técnico limitado o que dispongan de poco tiempo para interactuar con el ingeniero que recabará los requerimientos. Ninguna de estas posibilidades es deseable, pero todas son muy comunes y es frecuente verse forzado a trabajar con las restricciones impuestas por esta situación.

En las secciones que siguen se estudian las etapas requeridas para establecer las bases que permiten entender los requerimientos de software a fin de que el proyecto comience en forma tal que se mantenga avanzando hacia una solución exitosa.

PUNTO CLAVE

Un *participante* es cualquier persona que tenga interés directo o que se beneficie del sistema que se va a desarrollar.

5.2.1 Identificación de los participantes

Sommerville y Sawyer [Som97] definen *participante* como "cualquier persona que se beneficie en forma directa o indirecta del sistema en desarrollo". Ya se identificaron los candidatos habituales: gerentes de operaciones del negocio, gerentes de producto, personal de mercadotecnia, clientes internos y externos, usuarios finales, consultores, ingenieros de producto, ingenieros de software e ingenieros de apoyo y mantenimiento, entre otros. Cada participante tiene un punto de vista diferente respecto del sistema, obtiene distintos beneficios cuando éste se desarrolla con éxito y corre distintos riesgos si fracasa el esfuerzo de construcción.

---

7 Las herramientas mencionadas aquí no son obligatorias sino una muestra de las que hay en esta categoría. En la mayoría de casos, los nombres de las herramientas son marcas registradas por sus respectivos desarrolladores.

8 Este enfoque es ampliamente recomendable para proyectos que adoptan la filosofía de desarrollo de software ágil.

Durante la concepción, debe hacerse la lista de personas que harán aportes cuando se recaben los requerimientos (véase la sección 5.3). La lista inicial crecerá cuando se haga contacto con los participantes porque a cada uno se le hará la pregunta: "¿A quién más piensa que debe consultarse?"

Cita:

"Ponga a tres participantes en un cuarto y pregúnteles qué clase de sistema quieren. Es probable que escuche cuatro o más opiniones diferentes."

Anónimo

5.2.2 Reconocer los múltiples puntos de vista

Debido a que existen muchos participantes distintos, los requerimientos del sistema se explorarán desde muchos puntos de vista diferentes. Por ejemplo, el grupo de mercadotecnia se interesa en funciones y características que estimularán el mercado potencial, lo que hará que el nuevo sistema sea fácil de vender. Los gerentes del negocio tienen interés en un conjunto de características para que se elabore dentro del presupuesto y que esté listo para ocupar nichos de mercado definidos. Los usuarios finales tal vez quieran características que les resulten familiares y que sean fáciles de aprender y usar. Los ingenieros de software quizá piensen en funciones invisibles para los participantes sin formación técnica, pero que permitan una infraestructura que dé apoyo a funciones y características más vendibles. Los ingenieros de apoyo tal vez se centren en la facilidad del software para recibir mantenimiento.

Cada uno de estos integrantes (y otros más) aportará información al proceso de ingeniería de los requerimientos. A medida que se recaba información procedente de múltiples puntos de vista, los requerimientos que surjan tal vez sean inconsistentes o estén en conflicto uno con otro. Debe clasificarse toda la información de los participantes (incluso los requerimientos inconsistentes y conflictivos) en forma que permita a quienes toman las decisiones escoger para el sistema un conjunto de requerimientos que tenga coherencia interna.

5.2.3 Trabajar hacia la colaboración

Si en un proyecto de software hay involucrados cinco participantes, tal vez se tengan cinco (o más) diferentes opiniones acerca del conjunto apropiado de requerimientos. En los primeros capítulos se mencionó que, para obtener un sistema exitoso, los clientes (y otros participantes) debían colaborar entre sí (sin pelear por insignificancias) y con los profesionales de la ingeniería de software. Pero, ¿cómo se llega a esta colaboración?

El trabajo del ingeniero de requerimientos es identificar las áreas de interés común (por ejemplo, requerimientos en los que todos los participantes estén de acuerdo) y las de conflicto o incongruencia (por ejemplo, requerimientos que desea un participante, pero que están en conflicto con las necesidades de otro). Es la última categoría la que, por supuesto, representa un reto.

INFORMACIÓN

Uso de "puntos de prioridad"

Una manera de resolver requerimientos conflictivos y, al mismo tiempo, mejorar la comprensión de la importancia relativa de todos, es usar un esquema de "votación" con base en *puntos de prioridad*. Se da a todos los participantes cierto número de puntos de prioridad que pueden "gastarse" en cualquier número de requerimientos. Se presenta una lista de éstos y cada participante indica la importancia relativa de cada uno (desde su punto de vista) con la asignación de uno o más puntos de prioridad. Los puntos gastados ya no pueden utilizarse otra vez. Cuando un participante agota sus puntos de prioridad, ya no tiene la posibilidad de hacer algo con los requerimientos. El total de puntos asignados a cada requerimiento por los participantes da una indicación de la importancia general de cada requerimiento.

La colaboración no significa necesariamente que todos los requerimientos los defina un comité. En muchos casos, los participantes colaboran con la aportación de su punto de vista respecto de los requerimientos, pero un influyente "campeón del proyecto" (por ejemplo, el director

del negocio o un tecnólogo experimentado) toma la decisión final sobre los requerimientos que lo integrarán.

5.2.4 Hacer las primeras preguntas

Las preguntas que se hacen en la concepción del proyecto deben estar "libres del contexto" [Gau89]. El primer conjunto de ellas se centran en el cliente y en otros participantes, en las metas y beneficios generales. Por ejemplo, tal vez se pregunte:

Cita:

"Es mejor conocer algunas preguntas que todas las respuestas."

James Thurber

- ¿Quién está detrás de la solicitud de este trabajo?
- ¿Quién usará la solución?
- ¿Cuál será el beneficio económico de una solución exitosa?
- ¿Hay otro origen para la solución que se necesita?

Estas preguntas ayudan a identificar a todos los participantes con interés en el software que se va a elaborar. Además, las preguntas identifican el beneficio mensurable de una implementación exitosa y las posibles alternativas para el desarrollo de software personalizado.

Las preguntas siguientes permiten entender mejor el problema y hacen que el cliente exprese sus percepciones respecto de la solución:

? ¿Cuáles preguntas ayudarían a tener un entendimiento preliminar del problema?

- ¿Cuál sería una "buena" salida generada por una solución exitosa?
- ¿Qué problemas resolvería esta solución?
- ¿Puede mostrar (o describir) el ambiente de negocios en el que se usaría la solución?
- ¿Hay aspectos especiales del desempeño o restricciones que afecten el modo en el que se enfoque la solución?

Las preguntas finales se centran en la eficacia de la actividad de comunicación en sí. Gause y Weinberg [Gau89] las llaman "metapreguntas" y proponen la siguiente lista (abreviada):

Cita:

"El que hace una pregunta es tonto durante cinco minutos; el que no la hace será tonto para siempre."

Proverbio chino

- ¿Es usted la persona indicada para responder estas preguntas? ¿Sus respuestas son "oficiales"?
- ¿Mis preguntas son relevantes para el problema que se tiene?
- ¿Estoy haciendo demasiadas preguntas?
- ¿Puede otra persona dar información adicional?
- ¿Debería yo preguntarle algo más?

Estas preguntas (y otras) ayudarán a "romper el hielo" y a iniciar la comunicación, que es esencial para una indagación exitosa. Pero una reunión de preguntas y respuestas no es un enfoque que haya tenido un éxito apabullante. En realidad, la sesión de preguntas y respuestas sólo debe usarse para el primer encuentro y luego ser reemplazada por un formato de indagación de requerimientos que combine elementos de solución de problemas, negociación y especificación. En la sección 5.3 se presenta un enfoque de este tipo.

5.3 Indagación de los requerimientos

La indagación de los requerimientos (actividad también llamada *recabación de los requerimientos*) combina elementos de la solución de problemas, elaboración, negociación y especificación. A fin de estimular un enfoque colaborativo y orientado al equipo, los participantes trabajan juntos para identificar el problema, proponer elementos de la solución, negociar distintas visiones y especificar un conjunto preliminar de requerimientos para la solución [Zah90].[9]

9 En ocasiones se denomina a este enfoque *técnica facilitada de especificación de la aplicación* (TFEA).

5.3.1 Recabación de los requerimientos en forma colaborativa

Se han propuesto muchos enfoques distintos para recabar los requerimientos en forma colaborativa. Cada uno utiliza un escenario un poco diferente, pero todos son variantes de los siguientes lineamientos básicos:

¿Cuáles son los lineamientos básicos para conducir una reunión a fin de recabar los requerimientos en forma colaborativa?

- Tanto ingenieros de software como otros participantes dirigen o intervienen en las reuniones.
- Se establecen reglas para la preparación y participación.
- Se sugiere una agenda con suficiente formalidad para cubrir todos los puntos importantes, pero con la suficiente informalidad para que estimule el libre flujo de ideas.
- Un "facilitador" (cliente, desarrollador o participante externo) controla la reunión.
- Se utiliza un "mecanismo de definición" (que pueden ser hojas de trabajo, tablas sueltas, etiquetas adhesivas, pizarrón electrónico, grupos de conversación o foro virtual).

Cita:

"Dedicamos mucho tiempo —la mayor parte de todo el esfuerzo del proyecto— no a implementar o hacer pruebas, sino a tratar de decidir qué construir."

Brian Lawrence

La meta es identificar el problema, proponer elementos de la solución, negociar distintos enfoques y especificar un conjunto preliminar de requerimientos de la solución en una atmósfera que favorezca el logro de la meta. Para entender mejor el flujo de eventos conforme ocurren, se presenta un escenario breve que bosqueja la secuencia de hechos que llevan a la reunión para obtener requerimientos, a lo que sucede durante ésta y a lo que sigue después de ella.

Durante la concepción (véase la sección 5.2), hay preguntas y respuestas básicas que establecen el alcance del problema y la percepción general de lo que constituye una solución. Fuera de estas reuniones iniciales, el desarrollador y los clientes escriben una o dos páginas de "solicitud de producto".

Se selecciona un lugar, fecha y hora para la reunión, se escoge un facilitador y se invita a asistir a integrantes del equipo de software y de otras organizaciones participantes. Antes de la fecha de la reunión, se distribuye la solicitud de producto a todos los asistentes.

WebRef

La *solicitud conjunta de desarrollo* (SCD) es una técnica popular para recabar requerimientos. En la dirección **www.carolla.com/wp-jad.htm** se encuentra una buena descripción de ella.

Por ejemplo,[10] considere un extracto de una solicitud de producto escrita por una persona de mercadotecnia involucrada en el proyecto *CasaSegura*. Esta persona escribe la siguiente narración sobre la función de seguridad en el hogar que va a ser parte de *CasaSegura*:

> Nuestras investigaciones indican que el mercado para los sistemas de administración del hogar crece a razón de 40% anual. La primera función de *CasaSegura* que llevemos al mercado deberá ser la de seguridad del hogar. La mayoría de la gente está familiarizada con "sistemas de alarma", por lo que ésta deberá ser fácil de vender.
>
> La función de seguridad del hogar protegería, o reconocería, varias "situaciones" indeseables, como acceso ilegal, incendio y niveles de monóxido de carbono, entre otros. Emplearía sensores inalámbricos para detectar cada situación. Sería programada por el propietario y telefonearía en forma automática a una agencia de vigilancia cuando detectara una situación como las descritas.

CONSEJO

Si un sistema o producto servirá a muchos usuarios, asegúrese de que los requerimientos se obtengan de una franja representativa de ellos. Si sólo uno define todos los requerimientos, el riesgo de no aceptación es elevado.

En realidad, durante la reunión para recabar los requerimientos, otros contribuirían a esta narración y se dispondría de mucha más información. Pero aun con ésta habría ambigüedad, sería probable que existieran omisiones y ocurrieran errores. Por ahora bastará la "descripción funcional" anterior.

Mientras se revisa la solicitud del producto antes de la reunión, se pide a cada asistente que elabore una lista de objetos que sean parte del ambiente que rodeará al sistema, los objetos

10 Este ejemplo (con extensiones y variantes) se usa para ilustrar métodos importantes de la ingeniería de software en muchos de los capítulos siguientes. Como ejercicio, sería provechoso que el lector realizara su propia reunión para recabar requerimientos y que desarrollara un conjunto de listas para ella.

que producirá éste y los que usará para realizar sus funciones. Además, se solicita a cada asistente que haga otra lista de servicios (procesos o funciones) que manipulen o interactúen con los objetos. Por último, también se desarrollan listas de restricciones (por ejemplo, costo, tamaño, reglas del negocio, etc.) y criterios de desempeño (como velocidad y exactitud). Se informa a los asistentes que no se espera que las listas sean exhaustivas, pero sí que reflejen la percepción que cada persona tiene del sistema.

Cita:

"Los hechos no dejan de existir porque se les ignore."

Aldous Huxley

Entre los objetos descritos por *CasaSegura* tal vez estén incluidos el panel de control, detectores de humo, sensores en ventanas y puertas, detectores de movimiento, alarma, un evento (activación de un sensor), una pantalla, una computadora, números telefónicos, una llamada telefónica, etc. La lista de servicios puede incluir *configurar* el sistema, *preparar* la alarma, *vigilar* los sensores, *marcar* el teléfono, *programar* el panel de control y *leer* la pantalla (observe que los servicios actúan sobre los objetos). En forma similar, cada asistente desarrollará una lista de restricciones (por ejemplo, el sistema debe reconocer cuando los sensores no estén operando, debe ser amistoso con el usuario, debe tener una interfaz directa con una línea telefónica estándar, etc.) y de criterios de desempeño (un evento en un sensor debe reconocerse antes de un segundo, debe implementarse un esquema de prioridad de eventos, etcétera).

CONSEJO

Evite el impulso de desechar alguna idea de un cliente con expresiones como "demasiado costosa" o "imprácticа". La intención aquí es negociar una lista aceptable para todos. Para lograrlo, debe tenerse la mente abierta.

Las listas de objetos pueden adherirse a las paredes del cuarto con el empleo de pliegos de papel grandes o con láminas adhesivas, o escribirse en un tablero. Alternativamente, las listas podrían plasmarse en un boletín electrónico, sitio web interno o en un ambiente de grupo de conversación para revisarlas antes de la reunión. Lo ideal es que cada entrada de las listas pueda manipularse por separado a fin de combinar las listas o modificar las entradas y agregar otras. En esta etapa, están estrictamente prohibidos las críticas y el debate.

Una vez que se presentan las listas individuales acerca de un área temática, el grupo crea una lista, eliminando las entradas redundantes o agregando ideas nuevas que surjan durante el análisis, pero no se elimina ninguna. Después de crear listas combinadas para todas las áreas temáticas, sigue el análisis, coordinado por el facilitador. La lista combinada se acorta, se alarga o se modifica su redacción para que refleje de manera apropiada al producto o sistema que se va a desarrollar. El objetivo es llegar a un consenso sobre la lista de objetos, servicios, restricciones y desempeño del sistema que se va a construir.

En muchos casos, un objeto o servicio descrito en la lista requerirá mayores explicaciones. Para lograr esto, los participantes desarrollan *miniespecificaciones* para las entradas en las listas.[11] Cada miniespecificación es una elaboración de un objeto o servicio. Por ejemplo, la correspondiente al objeto **Panel de control** de *CasaSegura* sería así:

> El panel de control es una unidad montada en un muro, sus dimensiones aproximadas son de 9 por 5 pulgadas. Tiene conectividad inalámbrica con los sensores y con una PC. La interacción con el usuario tiene lugar por medio de un tablero que contiene 12 teclas. Una pantalla de cristal líquido de 3 por 3 pulgadas brinda retroalimentación al usuario. El software hace anuncios interactivos, como eco y funciones similares.

Las miniespecificaciones se presentan a todos los participantes para que sean analizadas. Se hacen adiciones, eliminaciones y otras modificaciones. En ciertos casos, el desarrollo de las miniespecificaciones descubrirá nuevos objetos, servicios o restricciones, o requerimientos de desempeño que se agregarán a las listas originales. Durante todos los análisis, el equipo debe posponer los aspectos que no puedan resolverse en la reunión. Se conserva una *lista de aspectos* para volver después a dichas ideas.

---

11 En vez de crear una miniespecificación, muchos equipos de software eligen desarrollar escenarios del usuario llamados *casos de uso*. Éstos se estudian en detalle en la sección 5.4 y en el capítulo 6.

CASASEGURA

Conducción de una reunión para recabar los requerimientos

La escena: Sala de juntas. Está en marcha la primera reunión para recabar los requerimientos.

Participantes: Jamie Lazar, integrante del equipo de software; Vinod Raman, miembro del equipo de software; Ed Robbins, miembro del equipo de software; Doug Miller, gerente de ingeniería de software; tres trabajadores de mercadotecnia; un representante de ingeniería del producto, y un facilitador.

La conversación:

Facilitador (apunta en un pizarrón): De modo que ésa es la lista actual de objetos y servicios para la función de seguridad del hogar.

Persona de mercadotecnia: Eso la cubre, desde nuestro punto de vista.

Vinod: ¿No dijo alguien que quería que toda la funcionalidad de *CasaSegura* fuera accesible desde internet? Eso incluiría la función de seguridad, ¿o no?

Persona de mercadotecnia: Sí, así es… tendremos que añadir esa funcionalidad y los objetos apropiados.

Facilitador: ¿Agrega eso algunas restricciones?

Jamie: Sí, tanto técnicas como legales.

Representante del producto: ¿Qué significa eso?

Jamie: Nos tendríamos que asegurar de que un extraño no pueda ingresar al sistema, desactivarlo y robar en el lugar o hacer algo peor. Mucha responsabilidad sobre nosotros.

Doug: Muy cierto.

Mercadotecnia: Pero lo necesitamos así… sólo asegúrense de impedir que ingrese un extraño.

Ed: Eso es más fácil de decir que de hacer.

Facilitador (interrumpe): No quiero que debatamos esto ahora. Anotémoslo como un aspecto y continuemos.

(Doug, que es el secretario de la reunión, toma debida nota.)

Facilitador: Tengo la sensación de que hay más por considerar aquí.

(El grupo dedica los siguientes 20 minutos a mejorar y aumentar los detalles de la función de seguridad del hogar.)

5.3.2 Despliegue de la función de calidad

El DFC define los requerimientos de forma que maximicen la satisfacción del cliente.

El *despliegue de la función de calidad* (DFC) es una técnica de administración de la calidad que traduce las necesidades del cliente en requerimientos técnicos para el software. El DFC "se concentra en maximizar la satisfacción del cliente a partir del proceso de ingeniería del software" [Zul92]. Para lograr esto, el DFC pone el énfasis en entender lo que resulta valioso para el cliente y luego despliega dichos valores en todo el proceso de ingeniería. El DFC identifica tres tipos de requerimientos [Zul92]:

Todos desean implementar muchos requerimientos emocionantes, pero hay que tener cuidado. Así es como empiezan a "quedar lisiados los requerimientos". Pero en contrapartida, los requerimientos emocionantes llevan a un avance enorme del producto…

Requerimientos normales. Objetivos y metas que se establecen para un producto o sistema durante las reuniones con el cliente. Si estos requerimientos están presentes, el cliente queda satisfecho. Ejemplos de requerimientos normales son los tipos de gráficos pedidos para aparecer en la pantalla, funciones específicas del sistema y niveles de rendimiento definidos.

Requerimientos esperados. Están implícitos en el producto o sistema y quizá sean tan importantes que el cliente no los mencione de manera explícita. Su ausencia causará mucha insatisfacción. Algunos ejemplos de requerimientos esperados son: fácil interacción humano/máquina, operación general correcta y confiable, y facilidad para instalar el software.

Requerimientos emocionantes. Estas características van más allá de las expectativas del cliente y son muy satisfactorias si están presentes. Por ejemplo, el software para un nuevo teléfono móvil viene con características estándar, pero si incluye capacidades inesperadas (como pantalla sensible al tacto, correo de voz visual, etc.) agrada a todos los usuarios del producto.

WebRef

En la dirección **www.qfdi.org** se encuentra información útil sobre el DFC.

Aunque los conceptos del DFC son aplicables en todo el proceso del software [Par96a], hay técnicas específicas de aquél que pueden aplicarse a la actividad de indagación de los requerimientos. El DFC utiliza entrevistas con los clientes, observación, encuestas y estudio de datos históricos (por ejemplo, reportes de problemas) como materia prima para la actividad de recabación

de los requerimientos. Después, estos datos se llevan a una tabla de requerimientos —llamada *tabla de la voz del cliente*— que se revisa con el cliente y con otros participantes. Luego se emplean varios diagramas, matrices y métodos de evaluación para extraer los requerimientos esperados y tratar de percibir requerimientos emocionantes [Aka04].

5.3.3 Escenarios de uso

A medida que se reúnen los requerimientos, comienza a materializarse la visión general de funciones y características del sistema. Sin embargo, es difícil avanzar hacia actividades más técnicas de la ingeniería de software hasta no entender cómo emplearán los usuarios finales dichas funciones y características. Para lograr esto, los desarrolladores y usuarios crean un conjunto de escenarios que identifican la naturaleza de los usos para el sistema que se va a construir. Los escenarios, que a menudo se llaman *casos de uso* [Jac92], proporcionan la descripción de la manera en la que se utilizará el sistema. Los casos de uso se estudian con más detalle en la sección 5.4.

CasaSegura

Desarrollo de un escenario preliminar de uso

La escena: Una sala de juntas, donde continúa la primera reunión para recabar los requerimientos.

Participantes: Jamie Lazar, integrante del equipo de software; Vinod Raman, miembro del equipo de software; Ed Robbins, miembro del equipo de software; Doug Miller, gerente de ingeniería de software; tres personas de mercadotecnia; un representante de ingeniería del producto, y un facilitador.

La conversación:

Facilitador: Hemos estado hablando sobre la seguridad para el acceso a la funcionalidad de *CasaSegura* si ha de ser posible el ingreso por internet. Me gustaría probar algo. Desarrollemos un escenario de uso para entrar a la función de seguridad.

Jamie: ¿Cómo?

Facilitador: Podríamos hacerlo de dos maneras, pero de momento mantengamos las cosas informales. Díganos (señala a una persona de mercadotecnia), ¿cómo visualiza el acceso al sistema?

Persona de mercadotecnia: Um… bueno, es la clase de cosa que haría si estuviera fuera de casa y tuviera que dejar entrar a alguien a ella —por ejemplo, una trabajadora doméstica o un técnico de reparaciones— que no tuviera el código de seguridad.

Facilitador (sonríe): Ésa es la razón por la que lo hace… dígame, ¿cómo lo haría en realidad?

Persona de mercadotecnia: Bueno… lo primero que necesitaría sería una PC. Entraría a un sitio web que mantendríamos para todos los usuarios de *CasaSegura*. Daría mi identificación de usuario y…

Vinod (interrumpe): La página web tendría que ser segura, encriptada, para garantizar que estuviéramos seguros y…

Facilitador (interrumpe): Ésa es buena información, Vinod, pero es técnica. Centrémonos en cómo emplearía el usuario final esta capacidad, ¿está bien?

Vinod: No hay problema.

Persona de mercadotecnia: Decía que entraría a un sitio web y daría mi identificación de usuario y dos niveles de clave.

Jamie: ¿Qué pasa si olvido mi clave?

Facilitador (interrumpe): Buena observación, Jamie, pero no entraremos a ella por ahora. Lo anotaremos y la llamaremos una *excepción*. Estoy seguro de que habrá otras.

Persona de mercadotecnia: Después de que introdujera las claves, aparecería una pantalla que representaría todas las funciones de *CasaSegura*. Seleccionaría la función de seguridad del hogar. El sistema pediría que verificara quién soy, pidiendo mi dirección o número telefónico o algo así. Entonces aparecería un dibujo del panel de control del sistema de seguridad y la lista de funciones que puede realizar —activar el sistema, desactivar el sistema o desactivar uno o más sensores—. Supongo que también me permitiría reconfigurar las zonas de seguridad y otras cosas como ésa, pero no estoy seguro.

(Mientras la persona de mercadotecnia habla, Doug toma muchas notas; esto forma la base para el primer escenario informal de uso. Alternativamente, hubiera podido pedirse a la persona de mercadotecnia que escribiera el escenario, pero esto se hubiera hecho fuera de la reunión.)

5.3.4 Indagación de los productos del trabajo

Los productos del trabajo generados como consecuencia de la indagación de los requerimientos variarán en función del tamaño del sistema o producto que se va a construir. Para la mayoría de sistemas, los productos del trabajo incluyen los siguientes:

¿Qué información se produce como consecuencia de recabar los requerimientos?

- Un enunciado de la necesidad y su factibilidad.
- Un enunciado acotado del alcance del sistema o producto.
- Una lista de clientes, usuarios y otros participantes que intervienen en la indagación de los requerimientos.
- Una descripción del ambiente técnico del sistema.
- Una lista de requerimientos (de preferencia organizados por función) y las restricciones del dominio que se aplican a cada uno.
- Un conjunto de escenarios de uso que dan perspectiva al uso del sistema o producto en diferentes condiciones de operación.
- Cualesquiera prototipos desarrollados para definir requerimientos.

Cada uno de estos productos del trabajo es revisado por todas las personas que participan en la indagación de los requerimientos.

5.4 DESARROLLO DE CASOS DE USO

En un libro que analiza cómo escribir casos de uso eficaces, Alistair Cockburn [Coc01b] afirma que "un caso de uso capta un contrato [...] [que] describe el comportamiento del sistema en distintas condiciones en las que el sistema responde a una petición de alguno de sus participantes [...]". En esencia, un caso de uso narra una historia estilizada sobre cómo interactúa un usuario final (que tiene cierto número de roles posibles) con el sistema en circunstancias específicas. La historia puede ser un texto narrativo, un lineamiento de tareas o interacciones, una descripción basada en un formato o una representación diagramática. Sin importar su forma, un caso de uso ilustra el software o sistema desde el punto de vista del usuario final.

PUNTO CLAVE

Los casos de uso se definen desde el punto de vista de un actor. Un actor es un papel que desempeñan las personas (usuarios) o los dispositivos cuando interactúan con el software.

El primer paso para escribir un caso de uso es definir un conjunto de "actores" que estarán involucrados en la historia. Los *actores* son las distintas personas (o dispositivos) que usan el sistema o producto en el contexto de la función y comportamiento que va a describirse. Los actores representan los papeles que desempeñan las personas (o dispositivos) cuando opera el sistema. Con una definición más formal, un *actor* es cualquier cosa que se comunique con el sistema o producto y que sea externo a éste. Todo actor tiene uno o más objetivos cuando utiliza el sistema.

Es importante notar que un actor y un usuario final no necesariamente son lo mismo. Un usuario normal puede tener varios papeles diferentes cuando usa el sistema, mientras que un actor representa una clase de entidades externas (gente, con frecuencia pero no siempre) que sólo tiene un papel en el contexto del caso de uso. Por ejemplo, considere al operador de una máquina (un usuario) que interactúa con la computadora de control de una celda de manufactura que contiene varios robots y máquinas de control numérico. Después de una revisión cuidadosa de los requerimientos, el software para la computadora de control requiere cuatro diferentes modos (papeles) para la interacción: modo de programación, modo de prueba, modo de vigilancia y modo de solución de problemas. Por tanto, es posible definir cuatro actores: programador, probador, vigilante y solucionador de problemas. En ciertos casos, el operador de la máquina desempeñará todos los papeles. En otros, distintas personas tendrán el papel de cada actor.

WebRef

Un artículo excelente sobre casos de uso puede descargarse desde la dirección **www.ibm.com/developerworks/webservices/library/codesign7.html**

Debido a que la indagación de los requerimientos es una actividad evolutiva, no todos los actores son identificados en la primera iteración. En ésta es posible identificar a los actores principales [Jac92], y a los secundarios cuando se sabe más del sistema. Los *actores principales* interactúan para lograr la función requerida del sistema y obtienen el beneficio previsto de éste. Trabajan con el software en forma directa y con frecuencia. Los *actores secundarios* dan apoyo al sistema, de modo que los primarios puedan hacer su trabajo.

Una vez identificados los actores, es posible desarrollar casos de uso. Jacobson [Jac92] sugiere varias preguntas[12] que debe responder un caso de uso:

¿Qué se necesita saber a fin de desarrollar un caso de uso eficaz?

- ¿Quién es el actor principal y quién(es) el(los) secundario(s)?
- ¿Cuáles son los objetivos de los actores?
- ¿Qué precondiciones deben existir antes de comenzar la historia?
- ¿Qué tareas o funciones principales son realizadas por el actor?
- ¿Qué excepciones deben considerarse al describir la historia?
- ¿Cuáles variaciones son posibles en la interacción del actor?
- ¿Qué información del sistema adquiere, produce o cambia el actor?
- ¿Tendrá que informar el actor al sistema acerca de cambios en el ambiente externo?
- ¿Qué información desea obtener el actor del sistema?
- ¿Quiere el actor ser informado sobre cambios inesperados?

En relación con los requerimientos básicos de *CasaSegura*, se definen cuatro actores: **propietario de la casa** (usuario), **gerente de arranque** (tal vez la misma persona que el **propietario de la casa**, pero en un papel diferente), **sensores** (dispositivos adjuntos al sistema) y **subsistema de vigilancia y respuesta** (estación central que vigila la función de seguridad de la casa de *CasaSegura*). Para fines de este ejemplo, consideraremos sólo al actor llamado **propietario de la casa**. Éste interactúa con la función de seguridad de la casa en varias formas distintas con el empleo del panel de control de la alarma o con una PC:

- Introduce una clave que permita todas las demás interacciones.
- Pregunta sobre el estado de una zona de seguridad.
- Interroga acerca del estado de un sensor.
- En una emergencia, oprime el botón de pánico.
- Activa o desactiva el sistema de seguridad.

Considerando la situación en la que el propietario de la casa usa el panel de control, a continuación se plantea el caso de uso básico para la activación del sistema:[13]

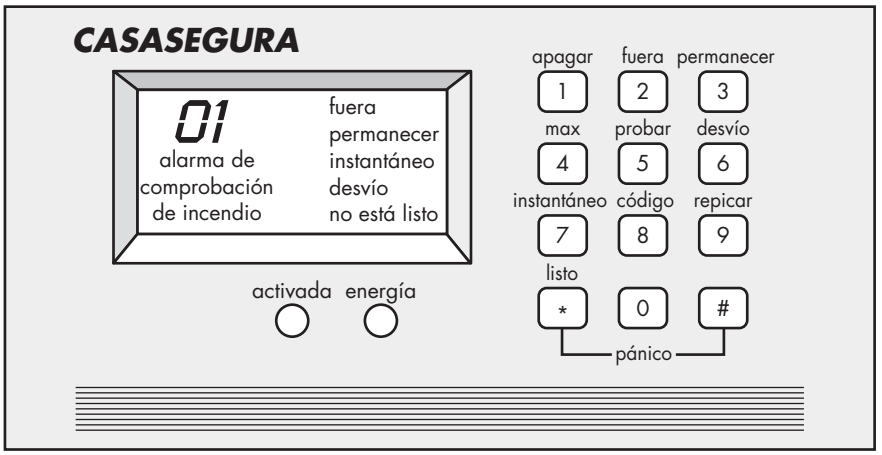
1. El propietario observa el panel de control de *CasaSegura* (véase la figura 5.1) para determinar si el sistema está listo para recibir una entrada. Si el sistema no está listo, se muestra el mensaje *no está listo* en la pantalla de cristal líquido y el propietario debe cerrar físicamente ventanas o puertas de modo que desaparezca dicho mensaje [el mensaje *no está listo* implica que un sensor está abierto; por ejemplo, que una puerta o ventana está abierta].
2. El propietario usa el teclado para introducir una clave de cuatro dígitos. La clave se compara con la que guarda el sistema como válida. Si la clave es incorrecta, el panel de control emitirá un sonido una vez y se reiniciará para recibir una entrada adicional. Si la clave es correcta, el panel de control espera otras acciones.
3. El propietario selecciona y teclea *permanecer* o *fuera* (véase la figura 5.1) para activar el sistema. La entrada *permanecer* activa sólo sensores perimetrales (se desactivan los sensores de detección de movimiento interior). La entrada *fuera* activa todos los sensores.
4. Cuando ocurre una activación, el propietario observa una luz roja de alarma.

---

12 Las preguntas de Jacobson se han ampliado para que den una visión más completa del contenido del caso de uso.

13 Observe que este caso de uso difiere de la situación en la que se accede al sistema a través de internet. En este caso, la interacción es por medio del panel de control y no con la interfaz de usuario gráfica (GUI) que se da cuando se emplea una PC.

FIGURA 5.1
Panel de control de *CasaSegura*



Es frecuente que los casos de uso se escriban de manera informal. Sin embargo, utilice el formato que se presenta aquí para asegurar que se incluyen todos los aspectos clave.

El caso de uso básico presenta una historia de alto nivel que describe la interacción entre el actor y el sistema.

En muchas circunstancias, los casos de uso son más elaborados a fin de que brinden muchos más detalles sobre la interacción. Por ejemplo, Cockburn [Coc01b] sugiere el formato siguiente para hacer descripciones detalladas de casos de uso:

Caso de uso: *IniciarVigilancia*

Actor principal: Propietario.

Objetivo en contexto: Preparar el sistema para que vigile los sensores cuando el propietario salga de la casa o permanezca dentro.

Precondiciones: El sistema se ha programado para recibir una clave y reconocer distintos sensores.

Disparador: El propietario decide "preparar" el sistema, por ejemplo, para que encienda las funciones de alarma.

Escenario:

1. Propietario: observa el panel de control
2. Propietario: introduce una clave
3. Propietario: selecciona "permanecer" o "fuera"
4. Propietario: observa una luz roja de alarma que indica que *CasaSegura* ha sido activada.

Excepciones:

1. El panel de control *no está listo*: el propietario verifica todos los sensores para determinar cuáles están abiertos; los cierra.
2. La clave es incorrecta (el panel de control suena una vez): el propietario introduce la clave correcta.
3. La clave no es reconocida: debe contactarse el subsistema de vigilancia y respuesta para reprogramar la clave.
4. Se elige *permanecer*: el panel de control suena dos veces y se enciende un letrero luminoso que dice *permanecer*; se activan los sensores del perímetro.
5. Se selecciona *fuera*: el panel de control suena tres veces y se enciende un letrero luminoso que dice *fuera*; se activan todos los sensores.

Prioridad: Esencial, debe implementarse

Cuándo estará disponible: En el primer incremento

Frecuencia de uso: Muchas veces por día

Canal para el actor: A través de la interfaz del panel de control

Actores secundarios: Técnico de apoyo, sensores

Canales para los actores secundarios:

Técnico de apoyo: línea telefónica

Sensores: interfaces cableadas y frecuencia de radio

Aspectos pendientes:

1. ¿Debe haber una forma de activar el sistema sin usar clave o con una clave abreviada?
2. ¿El panel de control debe mostrar mensajes de texto adicionales?
3. ¿De cuánto tiempo dispone el propietario para introducir la clave a partir del momento en el que se oprime la primera tecla?
4. ¿Hay una forma de desactivar el sistema antes de que se active en realidad?

Los casos de uso para otras interacciones de **propietario** se desarrollarían en una forma similar. Es importante revisar con cuidado cada caso de uso. Si algún elemento de la interacción es ambiguo, es probable que la revisión del caso de uso lo detecte.

CasaSegura

Desarrollo de un diagrama de caso de uso de alto nivel

La escena: Sala de juntas, continúa la reunión para recabar los requerimientos.

Participantes: Jamie Lazar, miembro del equipo de software; Vinod Roman, integrante del equipo de software; Ed Robbins, integrante del equipo de software; Doug Miller, gerente de ingeniería de software; tres miembros de mercadotecnia; un representante de ingeniería del producto; un facilitador.

La conversación:

Facilitador: Hemos pasado un buen tiempo hablando de la función de seguridad del hogar de *CasaSegura*. Durante el receso hice un diagrama de caso de uso para resumir los escenarios importantes que forman parte de esta función. Veámoslo.

(Todos los asistentes observan la figura 5.2.)

Jamie: Estoy aprendiendo la notación UML.[14] Veo que la función de seguridad del hogar está representada por el rectángulo grande con óvalos en su interior, ¿verdad? ¿Y los óvalos representan los casos de uso que hemos escrito?

Facilitador: Sí. Y las figuras pegadas representan a los actores —personas o cosas que interactúan con el sistema según los describe el caso de uso…—; ¡ah! usé el cuadrado para representar un actor que no es persona… en este caso, sensores.

Doug: ¿Es válido eso en UML?

Facilitador: La legalidad no es lo importante. El objetivo es comunicar información. Veo que usar una figura humana para representar un equipo sería erróneo. Así que adapté las cosas un poco. No pienso que genere problemas.

Vinod: Está bien, entonces tenemos narraciones de casos de uso para cada óvalo. ¿Necesitamos desarrollarlas con base en los formatos sobre los que he leído?

Facilitador: Es probable, pero eso puede esperar hasta que hayamos considerado otras funciones de *CasaSegura*.

Persona de mercadotecnia: Esperen, he estado observando este diagrama y de pronto me doy cuenta de que hemos olvidado algo.

Facilitador: ¿De verdad? Dime, ¿qué hemos olvidado?

(La reunión continúa.)

14 En el apéndice 1 se presenta un breve método de aprendizaje de UML para aquellos lectores que no estén familiarizados con dicha notación.

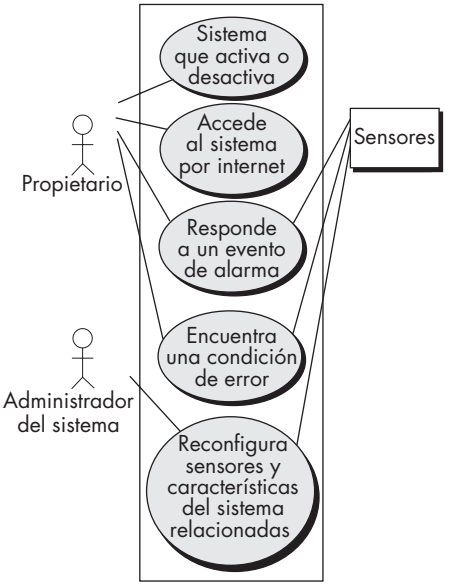


FIGURA 5.2 **Diagrama de caso de uso de UML para la función de seguridad del hogar de *CasaSegura***

HERRAMIENTAS DE SOFTWARE

Desarrollo de un caso de uso

Objetivo: Ayudar a desarrollar casos de uso proporcionando formatos y mecanismos automatizados para evaluar la claridad y consistencia.

Mecánica: La mecánica de las herramientas varía. En general, las herramientas para casos de uso dan formatos con espacios en blanco para ser llenados y crear así casos eficaces. La mayor parte de la funcionalidad de los casos de uso está incrustada en un conjunto más amplio de funciones de ingeniería de los requerimientos.

Herramientas representativas[15]

La gran mayoría de herramientas de análisis del modelado basadas en UML dan apoyo tanto de texto como gráfico para el desarrollo y modelado de casos de uso.

Objects by Design **(www.objectsbydesign.com/tools/umltools_byCompany.html)** proporciona vínculos exhaustivos con herramientas de este tipo.

5.5 ELABORACIÓN DEL MODELO DE LOS REQUERIMIENTOS[16]

El objetivo del modelo del análisis es describir los dominios de información, función y comportamiento que se requieren para un sistema basado en computadora. El modelo cambia en forma dinámica a medida que se aprende más sobre el sistema por construir, y otros participantes comprenden más lo que en realidad requieren. Por esa razón, el modelo del análisis es una fotografía de los requerimientos en cualquier momento dado. Es de esperar que cambie.

A medida que evoluciona el modelo de requerimientos, ciertos elementos se vuelven relativamente estables, lo que da un fundamento sólido para diseñar las tareas que sigan. Sin embargo, otros elementos del modelo son más volátiles, lo que indica que los participantes todavía no entienden bien los requerimientos para el sistema. En los capítulos 6 y 7 se presentan en

15 Las herramientas mencionadas aquí no son obligatorias, sino una muestra de las que hay en esta categoría. En la mayoría de casos, los nombres de las herramientas son marcas registradas por sus respectivos desarrolladores.

16 En este libro se usan como sinónimos las expresiones *modelar el análisis* y *modelar los requerimientos*. Ambos se refieren a representaciones de los dominios de la información, funcional y de comportamiento que describen los requerimientos del problema.

detalle el modelo del análisis y los métodos que se usan para construirlo. En las secciones siguientes se da un panorama breve.

5.5.1 Elementos del modelo de requerimientos

Hay muchas formas diferentes de concebir los requerimientos para un sistema basado en computadora. Algunos profesionales del software afirman que es mejor seleccionar un modo de representación (por ejemplo, el caso de uso) y aplicarlo hasta excluir a todos los demás. Otros piensan que es más benéfico usar cierto número de modos de representación distintos para ilustrar el modelo de requerimientos. Los modos diferentes de representación fuerzan a considerar los requerimientos desde distintos puntos de vista, enfoque que tiene una probabilidad mayor de detectar omisiones, inconsistencia y ambigüedades.

Los elementos específicos del modelo de requerimientos están determinados por el método de análisis de modelado (véanse los capítulos 6 y 7) que se use. No obstante, la mayoría de modelos tiene en común un conjunto de elementos generales.

Siempre es buena idea involucrar a los participantes. Una de las mejores formas de lograrlo es hacer que cada uno escriba casos de uso que narren el modo en el que se utilizará el software.

Elementos basados en el escenario. El sistema se describe desde el punto de vista del usuario con el empleo de un enfoque basado en el escenario. Por ejemplo, los casos de uso básico (véase la sección 5.4) y sus diagramas correspondientes de casos de uso (véase la figura 5.2) evolucionan hacia otros más elaborados que se basan en formatos. Los elementos del modelo de requerimientos basados en el escenario con frecuencia son la primera parte del modelo en desarrollo. Como tales, sirven como entrada para la creación de otros elementos de modelado. La figura 5.3 ilustra un diagrama de actividades UML[17] para indagar los requerimientos y representarlos con el empleo de casos de uso. Se aprecian tres niveles de elaboración que culminan en una representación basada en el escenario.

Una forma de aislar las clases es buscar sustantivos descriptivos en un caso de usuario expresado con texto. Al menos algunos de ellos serán candidatos cercanos. Sobre esto se habla más en el capítulo 8.

Elementos basados en clases. Cada escenario de uso implica un conjunto de objetos que se manipulan cuando un actor interactúa con el sistema. Estos objetos se clasifican en clases: conjunto de objetos que tienen atributos similares y comportamientos comunes. Por ejemplo, para ilustrar la clase **Sensor** de la función de seguridad de *Casa Segura* (véase la figura 5.4), puede utilizarse un diagrama de clase UML. Observe que el diagrama enlista los atributos de los sensores (por ejemplo, nombre, tipo, etc.) y las operaciones (por ejemplo, *identificar* y *permitir*) que se aplican para modificarlos. Además de los diagramas de clase, otros elementos de modelado del análisis ilustran la manera en la que las clases colaboran una con otra y las relaciones e interacciones entre ellas. Esto se analiza con más detalle en el capítulo 7.

Elementos de comportamiento. El comportamiento de un sistema basado en computadora tiene un efecto profundo en el diseño que se elija y en el enfoque de implementación que se aplique. Por tanto, el modelo de requerimientos debe proveer elementos de modelado que ilustren el comportamiento.

Un estado es un modo de comportamiento observable desde el exterior. Los estímulos externos ocasionan transiciones entre los estados.

El *diagrama de estado* es un método de representación del comportamiento de un sistema que ilustra sus estados y los eventos que ocasionan que el sistema cambie de estado. Un *estado* es cualquier modo de comportamiento observable desde el exterior. Además, el diagrama de estado indica acciones (como la activación de un proceso, por ejemplo) tomadas como consecuencia de un evento en particular.

Para ilustrar el uso de un diagrama de estado, considere el software incrustado dentro del panel de control de *CasaSegura* que es responsable de leer las entradas que hace el usuario. En la figura 5.5 se presenta un diagrama de estado UML simplificado.

Además de las representaciones de comportamiento del sistema como un todo, también es posible modelar clases individuales. Sobre esto se presentan más análisis en el capítulo 7.

---

17 En el apéndice 1 se presenta un instructivo breve sobre UML, para aquellos lectores que no estén familiarizados con dicha notación.

FIGURA 5.3

Diagramas de actividad del UML para indagar los requerimientos

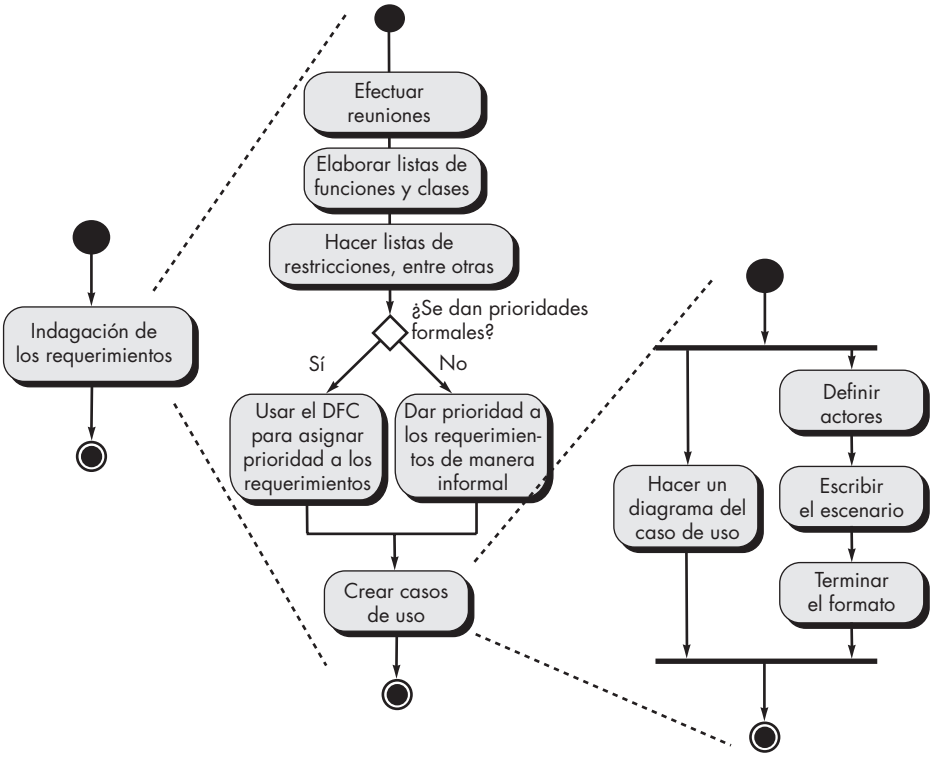


FIGURA 5.4

Diagrama de clase para un sensor

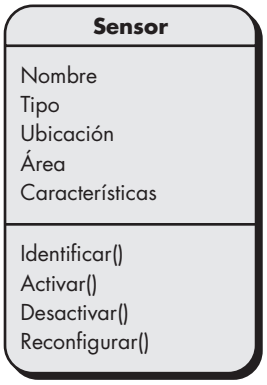
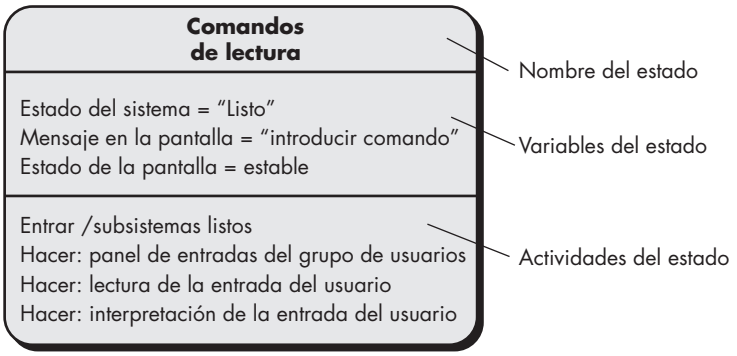


FIGURA 5.5

Notación UML del diagrama de estado



CasaSegura

Modelado preliminar del comportamiento

La escena: Sala de juntas, continúa la reunión de requerimientos.

Participantes: Jamie Lazar, integrante del equipo de software; Vinod Raman, miembro del equipo de software; Ed Robbins, integrante del equipo de software; Doug Miller, gerente de ingeniería de software; tres trabajadores de mercadotecnia; un representante de ingeniería del producto, y un facilitador.

La conversación:

Facilitador: Estamos por terminar de hablar sobre la funcionalidad de seguridad del hogar de *CasaSegura*. Pero antes, quisiera que analizáramos el comportamiento de la función.

Persona de mercadotecnia: No entiendo lo que quiere decir con *comportamiento*.

Ed (sonríe): Es cuando le das un "tiempo fuera" al producto si se porta mal.

Facilitador: No exactamente. Permítanme explicarlo.

(El facilitador explica al equipo encargado de recabar los requerimientos y los fundamentos de modelado del comportamiento.)

Persona de mercadotecnia: Esto parece un poco técnico. No estoy seguro de ser de ayuda aquí.

Facilitador: Seguro que lo serás. ¿Qué comportamiento se observa desde el punto de vista de un usuario?

Persona de mercadotecnia: Mmm... bueno, el sistema estará *vigilando* los sensores. *Leerá comandos* del propietario. *Mostrará* su estado.

Facilitador: ¿Ves?, lo puedes hacer.

Jamie: También estará *interrogando* a la PC para determinar si hay alguna entrada desde ella, por ejemplo, un acceso por internet o información sobre la configuración.

Vinod: Sí, en realidad, *configurar* el sistema es un estado por derecho propio.

Doug: Muchachos, lo hacen bien. Pensemos un poco más... ¿hay alguna forma de hacer un diagrama de todo esto?

Facilitador: Sí la hay, pero la dejaremos para la próxima reunión.

Elementos orientados al flujo. La información se transforma cuando fluye a través de un sistema basado en computadora. El sistema acepta entradas en varias formas, aplica funciones para transformarla y produce salidas en distintos modos. La entrada puede ser una señal de control transmitida por un transductor, una serie de números escritos con el teclado por un operador humano, un paquete de información enviado por un enlace de red o un archivo grande de datos recuperado de un almacenamiento secundario. La transformación quizá incluya una sola comparación lógica, un algoritmo numérico complicado o un enfoque de regla de inferencia para un sistema experto. La salida quizá encienda un diodo emisor de luz o genere un informe de 200 páginas. En efecto, es posible crear un modelo del flujo para cualquier sistema basado en computadora, sin importar su tamaño y complejidad. En el capítulo 7 se hace un análisis más detallado del modelado del flujo.

5.5.2 Patrones de análisis

Cualquiera que haya hecho la ingeniería de los requerimientos en varios proyectos de software ha observado que ciertos problemas son recurrentes en todos ellos dentro de un dominio de aplicación específico.[18] Estos *patrones de análisis* [Fow97] sugieren soluciones (por ejemplo, una clase, función o comportamiento) dentro del dominio de la aplicación que pueden volverse a utilizar cuando se modelen muchas aplicaciones.

Geyer-Schulz y Hahsler [Gey01] sugieren dos beneficios asociados con el uso de patrones de análisis:

> En primer lugar, los patrones de análisis aceleran el desarrollo de los modelos de análisis abstracto que capturan los principales requerimientos del problema concreto, debido a que proveen modelos de análisis reutilizables con ejemplos, así como una descripción de sus ventajas y limitaciones. En se-

18 En ciertos casos, los problemas vuelven a suceder sin importar el dominio de la aplicación. Por ejemplo, son comunes las características y funciones usadas para resolver problemas de la interfaz de usuario sin importar el dominio de la aplicación en consideración.

gundo lugar, los patrones de análisis facilitan la transformación del modelo de análisis en un modelo del diseño, sugiriendo patrones de diseño y soluciones confiables para problemas comunes.

Los patrones de análisis se integran en el modelo del análisis, haciendo referencia al nombre del patrón. También se guardan en un medio de almacenamiento de modo que los ingenieros de requerimientos usen herramientas de búsqueda para encontrarlos y aplicarlos. La información sobre el patrón de análisis (y otros tipos de patrones) se presenta en un formato estándar [Gey01][19] que se estudia con más detalle en el capítulo 12. En el capítulo 7 se dan ejemplos de patrones de análisis y más detalles de este tema.

5.6 Requerimientos de las negociaciones

Cita:

"Un compromiso es el arte de dividir un pastel en forma tal que todos crean que tienen el trozo mayor."

Ludwig Erhard

En un contexto ideal de la ingeniería de los requerimientos, las tareas de concepción, indagación y elaboración determinan los requerimientos del cliente con suficiente detalle como para avanzar hacia las siguientes actividades de la ingeniería de software. Desafortunadamente, esto rara vez ocurre. En realidad, se tiene que entrar en negociaciones con uno o varios participantes. En la mayoría de los casos, se pide a éstos que evalúen la funcionalidad, desempeño y otras características del producto o sistema, en contraste con el costo y el tiempo para entrar al mercado. El objetivo de esta negociación es desarrollar un plan del proyecto que satisfaga las necesidades del participante y que al mismo tiempo refleje las restricciones del mundo real (por ejemplo, tiempo, personas, presupuesto, etc.) que se hayan establecido al equipo del software.

Las mejores negociaciones buscan un resultado "ganar-ganar".[20] Es decir, los participantes ganan porque obtienen el sistema o producto que satisface la mayoría de sus necesidades y usted (como miembro del equipo de software) gana porque trabaja con presupuestos y plazos realistas y asequibles.

Boehm [Boe98] define un conjunto de actividades de negociación al principio de cada iteración del proceso de software. En lugar de una sola actividad de comunicación con el cliente, se definen las actividades siguientes:

1. Identificación de los participantes clave del sistema o subsistema.
2. Determinación de las "condiciones para ganar" de los participantes.

WebRef

Un artículo breve sobre la negociación para los requerimientos de software puede descargarse desde la dirección **www.alexander-egyed.com/publications/Software_Requirements_Negotiation-Some_Lessons_Learned.html**

Información

El arte de la negociación

Aprender a negociar con eficacia le servirá en su vida personal y técnica. Es útil considerar los lineamientos que siguen:

1. *Reconocer que no es una competencia.* Para tener éxito, ambas partes tienen que sentir que han ganado o logrado algo. Las dos tienen que llegar a un compromiso.
2. *Mapear una estrategia.* Decidir qué es lo que le gustaría lograr; qué quiere obtener la otra parte y cómo hacer para que ocurran las dos cosas.
3. *Escuchar activamente.* No trabaje en la formulación de su respuesta mientras la otra parte esté hablando. Escúchela. Es probable que obtenga conocimientos que lo ayuden a negociar mejor su posición.
4. *Centrarse en los intereses de la otra parte.* Si quiere evitar conflictos, no adopte posiciones inamovibles.
5. *No lo tome en forma personal.* Céntrese en el problema que necesita resolverse.
6. *Sea creativo.* Si están empantanados, no tenga miedo de pensar fuera de los moldes.
7. *Esté listo para comprometerse.* Una vez que se llegue a un acuerdo, no titubee; comprométase con él y cúmplalo.

19 En la bibliografía existen varias propuestas de formatos para patrones. Si el lector tiene interés, consulte [Fow97], [Gam95], [Yac03] y [Bus07], entre muchas otras fuentes.

20 Se han escrito decenas de libros acerca de las aptitudes para negociar (por ejemplo [Lew06], [Rai06] y [Fis06]). Es una de las aptitudes más importantes que pueda aprender. Lea alguno.

3. Negociación de las condiciones para ganar de los participantes a fin de reconciliarlas en un conjunto de condiciones ganar-ganar para todos los que intervienen (incluso el equipo de software).

La realización exitosa de estos pasos iniciales lleva a un resultado ganar-ganar, que se convierte en el criterio clave para avanzar hacia las siguientes actividades de la ingeniería de software.

CASASEGURA

El principio de una negociación

La escena: Oficina de Lisa Pérez, después de la primera reunión para recabar los requerimientos.

Participantes: Doug Miller, gerente de ingeniería de software, y Lisa Pérez, gerente de mercadotecnia.

La conversación:

Lisa: Pues escuché que la primera reunión salió realmente bien.

Doug: En realidad, sí. Enviaste buenos representantes... contribuyeron de verdad.

Lisa (sonríe): Sí; en realidad me dijeron que habían entrado y que no había sido una "actividad que les despejara la cabeza".

Doug (ríe): La próxima vez me aseguraré de quitarme la vena tecnológica... Mira, Lisa, creo que tenemos un problema para llegar a toda esa funcionalidad del sistema de seguridad para el hogar en las fechas que propone tu dirección. Sé que aún es temprano, pero hice un poco de planeación sobre las rodillas y...

Lisa (con el ceño fruncido): Lo debemos tener para esa fecha, Doug. ¿De qué funcionalidad hablas?

Doug: Supongo que podemos tener la funcionalidad completa en la fecha establecida, pero tendríamos que retrasar el acceso por internet hasta el segundo incremento.

Lisa: Doug, es el acceso por internet lo que da a *CasaSegura* su "súper" atractivo. Toda nuestra campaña de publicidad va a girar alrededor de eso. Lo tenemos que tener…

Doug: Entiendo la situación, de verdad. El problema es que para dar acceso por internet tendríamos que tener un sitio web por completo seguro y en operación. Esto requiere tiempo y personal. También tenemos que elaborar mucha funcionalidad adicional en la primera entrega… no creo que podamos hacerlo con los recursos que tenemos.

Lisa (todavía frunce el ceño): Ya veo, pero tienes que imaginar una manera de hacerlo. Tiene importancia crítica para las funciones de seguridad del hogar y también para otras… éstas podrían esperar hasta las siguientes entregas… estoy de acuerdo con eso.

Lisa y Doug parecen estar en suspenso, pero todavía deben negociar una solución a este problema. ¿Pueden "ganar" los dos en este caso? Si usted fuera el mediador, ¿qué sugeriría?

5.7 VALIDACIÓN DE LOS REQUERIMIENTOS

A medida que se crea cada elemento del modelo de requerimientos, se estudia para detectar inconsistencias, omisiones y ambigüedades. Los participantes asignan prioridades a los requerimientos representados por el modelo y se agrupan en paquetes de requerimientos que se implementarán como incrementos del software. La revisión del modelo de requerimientos aborda las preguntas siguientes:

? Cuando se revisan los requerimientos, ¿qué preguntas deben plantearse?

- ¿Es coherente cada requerimiento con los objetivos generales del sistema o producto?
- ¿Se han especificado todos los requerimientos en el nivel apropiado de abstracción? Es decir, ¿algunos de ellos tienen un nivel de detalle técnico que resulta inapropiado en esta etapa?
- El requerimiento, ¿es realmente necesario o representa una característica agregada que tal vez no sea esencial para el objetivo del sistema?
- ¿Cada requerimiento está acotado y no es ambiguo?
- ¿Tiene atribución cada requerimiento? Es decir, ¿hay una fuente (por lo general una individual y específica) clara para cada requerimiento?
- ¿Hay requerimientos en conflicto con otros?

- ¿Cada requerimiento es asequible en el ambiente técnico que albergará el sistema o producto?
- Una vez implementado cada requerimiento, ¿puede someterse a prueba?
- El modelo de requerimientos, ¿refleja de manera apropiada la información, la función y el comportamiento del sistema que se va a construir?
- ¿Se ha "particionado" el modelo de requerimientos en forma que exponga información cada vez más detallada sobre el sistema?
- ¿Se ha usado el patrón de requerimientos para simplificar el modelo de éstos? ¿Se han validado todos los patrones de manera apropiada? ¿Son consistentes todos los patrones con los requerimientos del cliente?

Éstas y otras preguntas deben plantearse y responderse para garantizar que el modelo de requerimientos es una reflexión correcta sobre las necesidades del participante y que provee un fundamento sólido para el diseño.

5.8 Resumen

Las tareas de la ingeniería de requerimientos se realizan para establecer un fundamento sólido para el diseño y la construcción. La ingeniería de requerimientos ocurre durante las actividades de comunicación y modelado que se hayan definido para el proceso general del software. Los miembros del equipo de software llevan a cabo siete funciones de ingeniería de requerimientos: concepción, indagación, elaboración, negociación, especificación, validación y administración.

En la concepción del proyecto, los participantes establecen los requerimientos básicos del problema, definen las restricciones generales del proyecto, así como las características y funciones principales que debe presentar el sistema para cumplir sus objetivos. Esta información se mejora y amplía durante la indagación, actividad en la que se recaban los requerimientos y que hace uso de reuniones que lo facilitan, DFC y el desarrollo de escenarios de uso.

La elaboración amplía aún más los requerimientos en un modelo: una colección de elementos basados en escenarios, clases y comportamiento, y orientados al flujo. El modelo hace referencia a patrones de análisis: soluciones para problemas de análisis que se ha observado que son recurrentes en diferentes aplicaciones.

Conforme se identifican los requerimientos y se crea su modelo, el equipo de software y otros participantes negocian la prioridad, la disponibilidad y el costo relativo de cada requerimiento. Además, se valida cada requerimiento y su modelo como un todo comparado con las necesidades del cliente a fin de garantizar que va a construirse el sistema correcto.

Problemas y puntos por evaluar

5.1. ¿Por qué muchos desarrolladores de software no ponen atención suficiente a la ingeniería de requerimientos? ¿Existen algunas circunstancias que puedan ignorarse?

5.2. El lector tiene la responsabilidad de indagar los requerimientos de un cliente que dice estar demasiado ocupado para tener una reunión. ¿Qué debe hacer?

5.3. Analice algunos de los problemas que ocurren cuando los requerimientos deben indagarse para tres o cuatro clientes distintos.

5.4. ¿Por qué se dice que el modelo de requerimientos representa una fotografía instantánea del sistema en el tiempo?

5.5. Suponga que ha convencido al cliente (es usted muy buen vendedor) para que esté de acuerdo con todas las demandas que usted hace como desarrollador. ¿Eso lo convierte en un gran negociador? ¿Por qué?

5.6. Desarrolle al menos tres "preguntas libres de contexto" adicionales que podría plantear a un participante durante la concepción.

5.7. Desarrolle un "kit" para recabar requerimientos. Debe incluir un conjunto de lineamientos a fin de llevar a cabo la reunión para recabar requerimientos y los materiales que pueden emplearse para facilitar la creación de listas y otros objetos que ayuden a definir los requerimientos.

5.8. Su profesor formará grupos de cuatro a seis estudiantes. La mitad de ellos desempeñará el papel del departamento de mercadotecnia y la otra mitad adoptará el del equipo para la ingeniería de software. Su trabajo es definir los requerimientos para la función de seguridad de *CasaSegura* descrita en este capítulo. Efectúe una reunión para recabar los requerimientos con el uso de los lineamientos presentados en este capítulo.

5.9. Desarrolle un caso de uso completo para una de las actividades siguientes:

a) Hacer un retiro de efectivo en un cajero automático.
b) Usar su tarjeta de crédito para pagar una comida en un restaurante.
c) Comprar acciones en la cuenta en línea de una casa de bolsa.
d) Buscar libros (sobre un tema específico) en una librería en línea.
e) La actividad que especifique su profesor.

5.10. ¿Qué representan las "excepciones" en un caso de uso?

5.11. Describa con sus propias palabras lo que es un *patrón de análisis*.

5.12. Con el formato presentado en la sección 5.5.2, sugiera uno o varios patrones de análisis para los siguientes dominios de aplicación:

a) Software de contabilidad.
b) Software de correo electrónico.
c) Navegadores de internet.
d) Software de procesamiento de texto.
e) Software para crear un sitio web.
f) El dominio de aplicación que diga su profesor.

5.13. ¿Qué significa *ganar-ganar* en el contexto de una negociación durante la actividad de ingeniería de los requerimientos?

5.14. ¿Qué piensa que pasa cuando la validación de los requerimientos detecta un error? ¿Quién está involucrado en su corrección?

Lecturas adicionales y fuentes de información

La ingeniería de requerimientos se estudia en muchos libros debido a su importancia crítica para la creación exitosa de cualquier sistema basado en computadoras. Hood *et al.* (*Requirements Management*, Springer, 2007) analizan varios aspectos de la ingeniería de los requerimientos que incluyen tanto la ingeniería de sistemas como la de software. Young (*The Requirements Engineering Handbook*, Artech House Publishers, 2007) presenta un análisis profundo de las tareas de la ingeniería de requerimientos. Wiegers (*More About Software Requirements*, Microsoft Press, 2006) menciona muchas técnicas prácticas para recabar y administrar los requerimientos. Hull *et al.* (*Requirements Engineering*, 2a. ed., Springer-Verlag, 2004), Bray (*An Introduction to Requirements Engineering*, Addison-Wesley, 2002), Arlow (*Requirements Engineering*, Addison-Wesley, 2001), Gilb (*Requirements Engineering*, Addison-Wesley, 2000), Graham (*Requirements Engineering and Rapid Development*, Addison-Wesley, 1999) y Sommerville y Kotonya (*Requirement Engineering: Processes and Techniques*, Wiley, 1998) son sólo algunos de los muchos libros dedicados al tema. Gottesdiener (*Requirements by Collaboration: Workshops for Defining Needs*, Addison-Wesley, 2002) proporciona una guía útil para quienes deben generar un ambiente de colaboración a fin de recabar los requerimientos con los participantes.

Lauesen (*Software Requirements: Styles and Techniques*, Addison-Wesley, 2002) presenta una recopilación exhaustiva de los métodos y notación para el análisis de requerimientos. Weigers (*Software Requirements*, Microsoft Press, 1999) y Leffingwell *et al.* (*Managing Software Requirements: A Use Case Approach*, 2a. ed., Addison-Wesley, 2003) presentan una colección útil de las mejores prácticas respecto de los requerimientos y sugieren lineamientos prácticos para la mayoría de los aspectos del proceso de su ingeniería.

En Withall (*Software Requirement Patterns*, Microsoft Press, 2007) se describe la ingeniería de requerimientos desde un punto de vista basado en los patrones. Ploesch (*Assertions, Scenarios and Prototypes*, Springer-Verlag, 2003) analiza técnicas avanzadas para desarrollar requerimientos de software. Windle y Abreo (*Software Requirements Using the Unified Process*, Prentice-Hall, 2002) estudian la ingeniería de los requerimientos en el contexto del proceso unificado y la notación UML. Alexander y Steven (*Writing Better Requirements*, Addison-Wesley, 2002) presentan un conjunto abreviado de lineamientos para escribir requerimientos claros, representarlos como escenarios y revisar el resultado final.

Es frecuente que el modelado de un caso de uso sea el detonante para crear todos los demás aspectos del modelo de análisis. El tema lo estudian mucho Rosenberg y Stephens (*Use Case Driven Object Modeling with UML: Theory and Practice*, Apress, 2007), Denny (*Succeeding with Use Cases: Working Smart to Deliver Quality*, Addison-Wesley, 2005), Alexander y Maiden (eds.) (*Scenarios, Stories, Use Cases: Through the Systems Development Life-Cycle*, Wiley, 2004), Leffingwell *et al.* (*Managing Software Requirements: A Use Case Approach*, 2a. ed., Addison-Wesley, 2003) presentan una colección útil de las mejores prácticas sobre los requerimientos. Bittner y Spence (*Use Case Modeling*, Addison-Wesley, 2002), Cockburn [Coc01], Armour y Miller (*Advanced Use Cases Modeling: Software Systems*, Addison-Wesley, 2000) y Kulak *et al.* (*Use Cases: Requirements in Context*, Addison-Wesley, 2000) estudian la obtención de requerimientos con énfasis en el modelado del caso de uso.

En internet hay una variedad amplia de fuentes de información acerca de la ingeniería y análisis de los requerimientos. En el sitio web del libro, **www.mhhe.com/engcs/compsi/pressman/professional/olc/ser.htm**, se halla una lista actualizada de referencias en web que son relevantes para la ingeniería y análisis de los requerimientos.