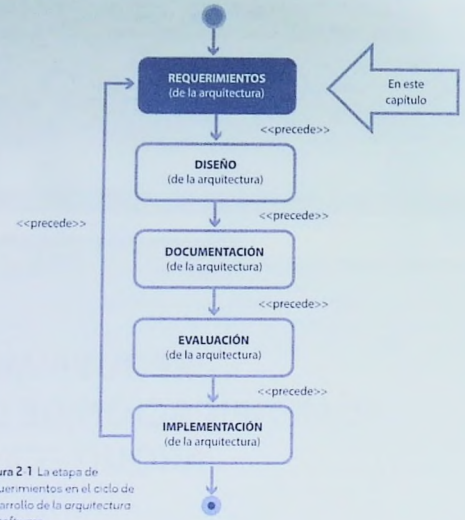
CAPÍTULO 2

REQUERIMIENTOS: IDENTIFICACIÓN DE *DRIVERS* ARQUITECTÓNICOS

En el capítulo anterior mencionamos que el desarrollo de la *arquitectura de software* involucra un proceso de cinco etapas. A partir de este capítulo iniciamos con las descripciones correspondientes. Como muestra la figura 2-1, comenzamos con la etapa de requerimientos. A efecto de proveer el contexto necesario, primero describiremos el término requerimientos en el ámbito general de la *ingeniería de software*, y después nos enfocaremos en los aspectos particulares de la etapa, sus conceptos fundamentales, las fuentes de información de las cuales hace uso, así como algunos de los métodos más conocidos que se utilizan para llevar a cabo de manera sistemática sus actividades.



› Figura 2-1 La etapa de requerimientos en el ciclo de desarrollo de la *arquitectura de software*

2.1 REQUERIMIENTOS

En el primer capítulo se mencionó que la *arquitectura de software* tiene importancia especial porque la manera en que se estructura un sistema tiene impacto sobre la capacidad de este para alcanzar los objetivos de negocio. También se explicó que los objetivos de negocio del sistema son metas que busca alcanzar una organización y que motivan el desarrollo de un sistema.

Los objetivos de negocio del sistema pueden satisfacerse mediante comportamientos o características provistas por este. Por ejemplo, el objetivo de una organización que vende boletos de autobús, de ingresar a mercados internacionales, puede satisfacerse al permitir la compra en línea por medio de un portal y al tener facilidad para adaptar el sistema a diferentes idiomas, navegadores *web* y dispositivos móviles. Técnicamente hablando, y como se indica en la siguiente definición tomada de del libro *Software Requirements* (Wiegers, 2013), en el ámbito de la *ingeniería de software* las especificaciones de éstos comportamientos y características reciben el nombre de *requerimientos*.

> Requerimiento es una especificación que describe alguna funcionalidad, atributo o factor de calidad de un sistema de *software*. Puede describir también algún aspecto que restringe la forma en que se construye ese sistema.

En este contexto es importante considerar que, además de los comportamientos y características, la definición anterior indica también que es posible que los requerimientos describan aspectos que restringen el proceso para desarrollar un sistema de *software*. Por ejemplo, hay casos en que los sistemas deben implementarse bajo consideraciones especiales sobre la fecha de inicio o terminación, límites presupuestales o el uso de determinado tipo de tecnologías. En las siguientes secciones describiremos con mayor detalle la importancia de distinguir entre diferentes clases de requerimientos y la relación que estos tienen con el concepto de *drivers arquitectónicos*.

La identificación de los requerimientos ocurre durante el análisis, actividad técnica del desarrollo de sistemas que describimos en el capítulo anterior. En el contexto de la *ingeniería de software*, la ingeniería de requerimientos es la disciplina que engloba las actividades relacionadas con la obtención, análisis, documentación y validación de estos.

2.2 REQUERIMIENTOS CON DISTINTOS TIPOS Y NIVELES DE ABSTRACCIÓN

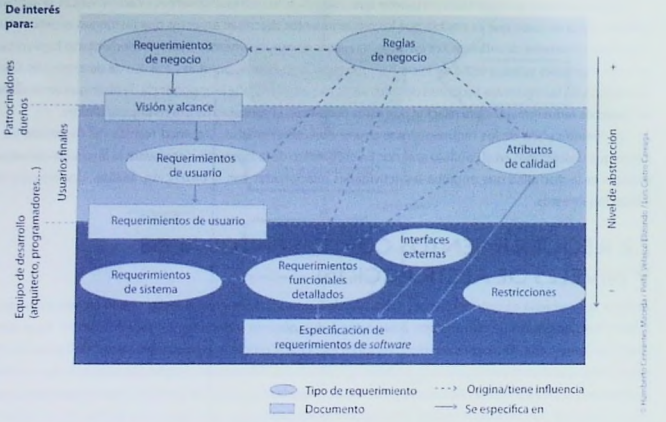
Aunque la ingeniería de requerimientos existe desde hace varios años, y a la fecha se han creado diversos métodos y herramientas que dan soporte a sus actividades, obtener y documentar requerimientos sigue siendo una tarea complicada. Recabarlos no solo consiste en escuchar al propietario o usuario final de un sistema, sino posteriormente escribir en una lista lo que él quiere que este haga. Además del propietario o usuario de un sistema, pueden existir otras personas interesadas en este; es importante reconocer correcta y oportunamente a todas ellas. Si bien los propietarios o los que hacen uso directo del sistema son más fáciles de identificar, quienes lo desarrollan, instalan y le dan mantenimiento, o bien, aquellos que patrocinaron su desarrollo (si no lo hicieron los dueños del sistema), son personas que también podrían estar interesadas en el sistema y que tal vez necesitarían tener requerimientos diferentes o adicionales a los de los propietarios o usuarios.

Debido a la heterogeneidad de los interesados en un sistema, no es extraño imaginar que la importancia que para alguno de ellos conlleva un requerimiento no sea la misma para los demás. Por ejemplo, la modificabilidad, la cual, como ya se mencionó, tiene que ver con qué tan simple es realizar cambios en un sistema, es un atributo de calidad más relevante probablemente para los desarrolladores que para los usuarios finales. De forma similar, incrementar los ingresos anuales es un requerimiento de interés para el dueño de un negocio, pero de menor importancia para esos clientes y desarrolladores. A efecto de facilitar su comprensión y manejo, es conveniente que los requerimientos sean clasificados en tipos y niveles de abstracción de acuerdo con su naturaleza.

En el contexto de la ingeniería de requerimientos se reconocen varias clases de estos, las cuales se ubican en distintos niveles de abstracción. La figura 2-2 las muestra en una versión adaptada de la figura original del modelo propuesto (Wiegers, 2013). En este modelo aparecen tres niveles. El nivel superior corresponde a tipos de requerimientos que describen aspectos del negocio. El nivel medio habla de cuestiones referidas a la interacción con los usuarios. Por último, el nivel inferior corresponde a los requerimientos que describen situaciones o elementos detallados que se necesitan para realizar el diseño del sistema. En el modelo también aparecen los documentos en dónde generalmente se especifican estos requerimientos.

La figura 2-3 muestra ejemplos de requerimientos en los niveles descritos anteriormente: a) requerimientos de negocio, que describen metas de una organización; b) requerimientos de usuario como casos de uso (Cockburn, 2001) o historias de usuario (Cohn, 2004) que describen los servicios que los usuarios pueden llevar a cabo por medio del sistema, o c) requerimientos funcionales detallados, como alguna especificación particular en el nivel de la interfaz de usuario.

En las secciones siguientes describimos en más detalle algunos tipos de requerimientos del modelo presentado antes.



› Figura 2-2 Tipos de requerimientos según Wiegers.

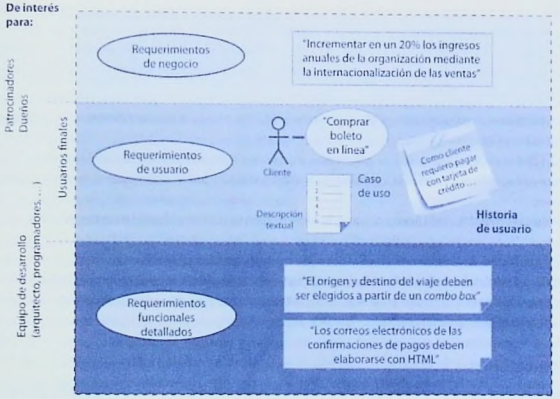
2.2.1 Requerimientos de usuario y requerimientos funcionales

Los requerimientos de usuario y los requerimientos funcionales especifican aspectos de carácter funcional sobre los servicios que pueden realizar los usuarios a través del sistema. Sin embargo, como se ilustra en la figura 2-3, se refieren a aspectos de diferente nivel de abstracción. Los requerimientos de usuario especifican servicios que por lo habitual dan soporte a procesos de negocio que los usuarios podrán llevar a cabo mediante el sistema, por ejemplo: "Comprar boleto (de autobús) en línea". Los funcionales describen detalles finos de diseño y/o implementación relacionados a los requerimientos de usuario, por ejemplo: "El origen y destino del viaje deben ser elegidos a partir de un *combo box*[1]". Ya mencionamos que los requerimientos de usuario se especifican por lo general mediante técnicas como casos de uso (en metodologías tradicionales) o historias de usuario (en metodologías ágiles).

En el conjunto de los requerimientos de usuario es posible distinguir dos tipos: primario y secundario. El primario describe servicios fundamentales que los usuarios desean llevar a cabo por medio del sistema, por ejemplo: "Comprar boleto (de autobús) en línea" Son fundamentales en tanto que soportan directamente los procesos de negocio clave de la organización. En contraste, el secundario describe acciones necesarias para dar soporte a la realización de los servicios especificados en los casos de uso o historias de usuario de los primarios. Para el ejemplo de caso de uso primario mencionado antes, casos de uso secundarios podrían ser: Acceder al sistema, Realizar alta o Enviar comprobante de compra.

Para facilitar la referencia a estos dos tipos, en lo sucesivo nos referiremos a ellos como requerimientos funcionales (que no deben confundirse con los funcionales detallados).

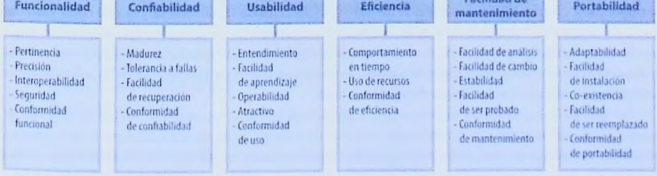
[1] Un *combo box* es una lista desplegable para hacer una selección en un conjunto de opciones predefinidas.



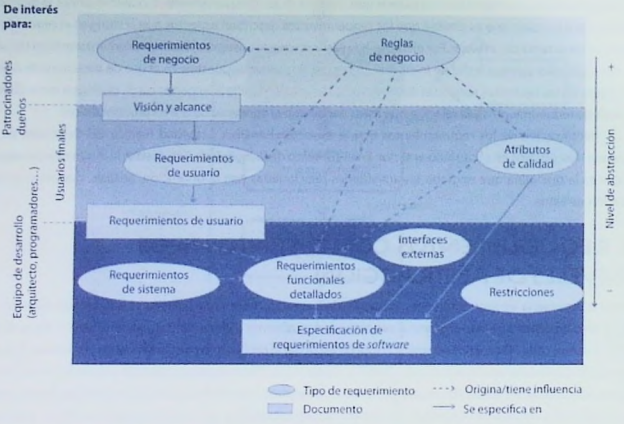
› Figura 2-3 Ejemplos de tipos específicos de requerimientos

2.2.2 Atributos de calidad

Los atributos de calidad especifican características útiles para establecer criterios sobre la calidad del sistema. Actualmente no existe un "listado único" de atributos de calidad relevantes para diseñar la *arquitectura de software*. Sin embargo, estándares para la evaluación de la calidad, como el ISO/IEC 9126 (*International Standard Organization*, 2001) o modelos de calidad como los definidos por McCall's (Pressman, 2001) o Boehm (Boehm, Brown y Lipow, 1976), pueden ser de utilidad para guiar al arquitecto durante la identificación y la especificación de este tipo de *drivers arquitectónicos*. La figura 2-4 presenta los atributos de calidad que considera el estándar ISO/IEC 9126.



›Figura 2-4 Atributos de calidad considerados por el estándar ISO/IEC 9126



› Figura 2-2 Tipos de requerimientos según Wiegers.

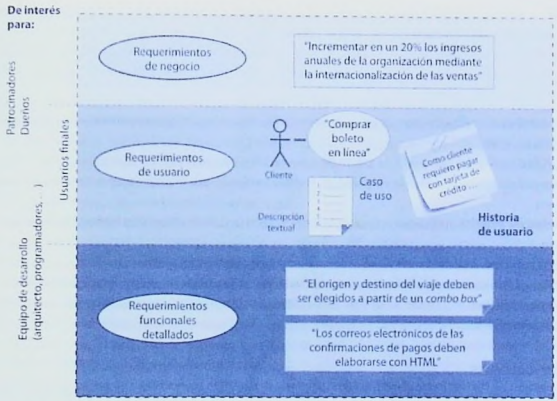
2.2.1 Requerimientos de usuario y requerimientos funcionales

Los requerimientos de usuario y los requerimientos funcionales especifican aspectos de carácter funcional sobre los servicios que pueden realizar los usuarios a través del sistema. Sin embargo, como se ilustra en la figura 2-3, se refieren a aspectos de diferente nivel de abstracción. Los requerimientos de usuario especifican servicios que por lo habitual dan soporte a procesos de negocio que los usuarios podrán llevar a cabo mediante el sistema, por ejemplo: "Comprar boleto (de autobús) en línea". Los funcionales describen detalles finos de diseño y/o implementación relacionados a los requerimientos de usuario, por ejemplo: "El origen y destino del viaje deben ser elegidos a partir de un *combo box*[1]". Ya mencionamos que los requerimientos de usuario se especifican por lo general mediante técnicas como casos de uso (en metodologías tradicionales) o historias de usuario (en metodologías ágiles).

En el conjunto de los requerimientos de usuario es posible distinguir dos tipos: primario y secundario. El primario describe servicios fundamentales que los usuarios desean llevar a cabo por medio del sistema, por ejemplo: "Comprar boleto (de autobús) en línea". Son fundamentales en tanto que soportan directamente los procesos de negocio clave de la organización. En contraste, el secundario describe acciones necesarias para dar soporte a la realización de los servicios especificados en los casos de uso o historias de usuario de los primarios. Para el ejemplo de caso de uso primario mencionado antes, casos de uso secundarios podrían ser: Acceder al sistema, Realizar alta o Enviar comprobante de compra.

Para facilitar la referencia a estos dos tipos, en lo sucesivo nos referiremos a ellos como requerimientos funcionales (que no deben confundirse con los funcionales detallados).

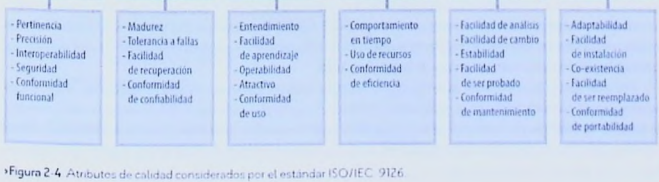
[1] Un *combo box* es una lista desplegable para hacer una selección en un conjunto de opciones predefinidas.



› Figura 2-3 Ejemplos de tipos específicos de requerimientos

2.2.2 Atributos de calidad

Los atributos de calidad especifican características útiles para establecer criterios sobre la calidad del sistema. Actualmente no existe un "listado único" de atributos de calidad relevantes para diseñar la *arquitectura de software*. Sin embargo, estándares para la evaluación de la calidad, como el ISO/IEC 9126 (*International Standard Organization*, 2001) o modelos de calidad como los definidos por McCall's (Pressman, 2001) o Boehm (Boehm, Brown y Lipow, 1976), pueden ser de utilidad para guiar al arquitecto durante la identificación y la especificación de este tipo de *drivers arquitectónicos*. La figura 2-4 presenta los atributos de calidad que considera el estándar ISO/IEC 9126.



›Figura 2-4 Atributos de calidad considerados por el estándar ISO/IEC 9126

información relevante para llevar a cabo la descomposición funcional del sistema y asignar estas funcionalidades a elementos específicos en la arquitectura.

Con ello, estos *drivers* se eligen considerando:

- Su relevancia en la satisfacción de los objetivos del negocio del sistema.
- La complejidad técnica que representa su implementación.
- El hecho de que representan algún escenario relevante para la arquitectura.

En la sección 2.1.1 del caso de estudio que conforma el apéndice de este libro se encuentra el modelo de casos de uso el cual incluye, entre otros, Registrarse en el sistema, Consultar corridas y Comprar boleto. Considerando los criterios anteriores, en la sección 2.1.2 se describe cómo los casos de uso primarios Consultar corridas y Comprar boleto son elegidos como *drivers* funcionales.

2.3.2 *Drivers* de atributos de calidad

Por lo habitual, todos los requerimientos de atributos de calidad son *drivers* de la arquitectura, independientemente de su prioridad. Sin embargo como el tiempo para realizar el diseño arquitectónico es acotado, para producir un diseño inicial es preferible considerar nada más un subconjunto de los requerimientos de atributos de calidad.

Al igual que con los *drivers* funcionales, los requerimientos de esta clase se eligen por lo habitual considerando estos criterios:

- Su relevancia en la satisfacción de los objetivos del negocio del sistema, es decir, su importancia para el cliente.
- La complejidad técnica que representa su implementación, esta es su importancia para el arquitecto.

Teniendo en cuenta los criterios anteriores, en la sección 2.2 del apéndice se ejemplifica y discute la elección de atributos de calidad como desempeño y disponibilidad como *drivers* del caso de estudio de este libro.

2.3.3 *Drivers* de restricciones

Como se mencionó antes, en contraste con los *drivers* funcionales y de atributos de calidad, los requerimientos de este tipo no tienen prioridad. Por esta razón, todas las restricciones son *drivers* de la arquitectura.

Como ejemplos en la sección 2.3 del apéndice, se observan algunas restricciones técnicas y administrativas asociadas al sistema del caso de estudio.

2.3.4 Otra información

Aun cuando los *drivers* descritos anteriormente especifican gran parte de la información necesaria para comenzar a diseñar un sistema, existen otros requerimientos que también podrían ser considerados por el arquitecto. Es el caso de las reglas de negocio y las interfaces externas, que podrían ser relevantes si estos determinan, de algún modo, la forma de las estructuras o los elementos arquitectónicos.

Como ejemplo retomemos el caso de uso: "Comprar boleto (de autobús) en línea". Si hay una regla de negocio, la cual indique que una vez seleccionada la corrida y los asientos, el cliente tiene hasta 24 horas para reservar y, eventualmente, completar la compra, esto podría influir en el arquitecto a usar una infraestructura de transacciones autentificadas que tienen duraciones de sesión definidas.

De modo similar, la historia de usuario: "Como cliente requiero pagar con tarjeta de crédito", podría necesitar la comunicación con un componente externo de negocios electrónicos que se encargue de los pagos. Conocer el protocolo de comunicación utilizado por el componente externo de procesamiento de pagos es importante para definir las interfaces de los componentes del sistema que se comunican con él.

2.3.5 Influencia de los *drivers arquitectónicos* en el diseño de la arquitectura

Aun cuando la información de los *drivers* influye en las decisiones de diseño tomadas por el arquitecto durante el diseño de la arquitectura, tal influencia no está igualmente repartida en todos ellos. Los de requerimientos funcionales pueden a menudo ser implementados fácilmente si se consideran en un contexto "aislado". No obstante, en un ámbito en donde también existen restricciones y expectativas sobre atributos de calidad relacionadas a ellos, implementarlos puede no ser una tarea trivial.

Por ejemplo, si se toma el caso de uso: "Comprar boleto (de autobús) en línea" como un *driver* de requerimiento funcional, en términos generales este podría ser especificado en función de la secuencia de acciones siguiente: 1) mostrar al usuario las opciones de viajes disponibles para el destino de su interés; 2) recibir del usuario la opción seleccionada; 3) formalizar la compra mediante el cobro del boleto, y 4) actualizar el número de asientos disponibles en el viaje correspondiente.

Hablando en términos de *arquitectura de software*, una alternativa válida sería asignar estas cuatro acciones a un solo componente, considerando las interfaces necesarias para la modificación de información en las bases de datos correspondientes, así como la comunicación con el componente externo que procesará el cobro del boleto. Si este requerimiento se presenta en un contexto en donde además se necesita que el procesamiento de la compra deba realizarse en un tiempo no mayor a cinco segundos, es evidente que la estrategia de diseño descrita no es la mejor.

Asignar estas cuatro acciones a un solo componente podría propiciar "cuellos de botella" en horarios de mayor uso del sistema y, por consiguiente, afectar el tiempo de respuesta. De forma similar, tal diseño propicia que el componente sea un punto de vulnerabilidad. Si este falla, el tiempo requerido para restaurarlo impedirá que la compra se complete en un tiempo no mayor a cinco segundos. Para tal escenario sería más adecuado un diseño que considere un particionamiento en donde algunas de las funciones independientes puedan asignarse a diferentes componentes, y algunos de estos, a su vez, pudieran estar replicados.

En la actualidad, muchos productos de *software* comerciales ofrecen soporte para la implementación de esquemas de replicación, de modo que el arquitecto podría considerarlos durante el diseño. Sin embargo, si en el proyecto existen restricciones presupuestales respecto del tipo de soluciones de soporte que pueden ser adquiridas, el uso de estos productos podría limitarse a los que sean gratuitos o tengan un costo bajo.

Lo anterior ejemplifica que aunque la funcionalidad es un aspecto fundamental en todos los sistemas de *software*, no es lo único que determina la forma de la arquitectura. Todo sistema tiene asociados, en mayor o menor medida, restricciones y requerimientos de atributos de calidad. Minimizar o ignorar la importancia de esta información durante el desarrollo de la arquitectura es sumamente riesgoso porque puede ser un factor importante del fracaso de un proyecto de *software*.

Es importante recordar que:

> del conjunto de *drivers arquitectónicos*, los atributos de calidad y las restricciones son los requerimientos que tienen mayor influencia sobre el diseño de la *arquitectura de software*.

2.4 FUENTES DE INFORMACIÓN PARA LA EXTRACCIÓN DE *DRIVERS ARQUITECTÓNICOS*

En esta sección presentamos en mayor detalle algunos de los artefactos que contienen información para extraer los *drivers arquitectónicos*.

2.4.1 Documento de visión y alcance

En términos generales, el documento de visión y alcance contiene información referente a por qué desarrollar un sistema, y al alcance de este, mediante descripciones textuales acerca de su contexto, las necesidades que resuelve, la descripción de sus usuarios, entre otras. El modelo presentado en la figura 2-2 considera este documento e indica que los requerimientos de negocio del sistema están especificados ahí. El documento de visión y alcance sirve de base y/o resulta complementario durante la identificación de *drivers* porque es útil, por ejemplo, al realizar actividades de priorización de estos respecto de su relevancia en la satisfacción de los objetivos de negocio del sistema.

Si bien en la práctica hay variaciones en el número de elementos de información contenidos en el documento de visión y alcance, gran parte de ellos puede ser utilizada como datos de entrada de varios métodos de la etapa de requerimientos. La figura 2-5 presenta, según su autor (Wiegers, 2013) la estructura y elementos de información sugeridos para el documento. En la sección 1 del apéndice se presenta un ejemplo de documento de visión y alcance con una versión simplificada del formato mostrado en la figura 2-5.

1. **Requerimientos de negocio**
   1.1 Antecedentes
   1.2 Oportunidad de negocio
   1.3 Objetivos de negocio y criterios de éxito
   1.4 Necesidades del cliente o del mercado
   1.5 Riesgos de negocio
2. **Visión de la solución**
   2.1 Enunciado de visión
   2.2 Características principales
   2.3 Suposiciones y dependencias
3. **Alcance y limitaciones**
   3.1 Alcance de la versión inicial
   3.2 Alcance de las versiones subsecuentes
   3.3 Limitaciones y exclusiones
4. **Contexto de negocio**
   4.1 Perfiles de los interesados en el sistema
   4.2 Prioridades del proyecto
   4.3 Ambiente de operación

› **Figura 2-5** Ejemplo de estructura y elementos de información sugeridos para el documento de visión y alcance

2.4.2 Documento de requerimientos de usuario

Los casos de uso y las historias de usuario son elementos utilizados para especificar necesidades de los clientes, es decir, para especificar los servicios que estos pueden realizar por medio del sistema (véase la figura 2-3). En el contexto de la ingeniería de requerimientos, en el modelo de la figura 2-2, tal información es parte del documento de requerimientos de usuario. Es válido que este no se genere de manera formal, sobre todo en proyectos de desarrollo que usan metodologías ágiles. Lo importante en esta etapa es que la información esté disponible para el arquitecto.

Los casos de uso (Cockburn, 2001) , además de tener una representación visual en forma de elipse en el diagrama de casos de uso, incluyen una descripción textual que detalla la secuencia de interacciones entre el sistema y los actores. Los elementos en la descripción textual pueden variar, aunque por lo habitual se incluye el nombre del caso de uso, actores involucrados, objetivos de negocio relacionados, precondiciones y poscondiciones, así

como descripciones de los flujos de interacciones principales, alternativas y de error. En conjunto al diagrama de casos de uso y las descripciones textuales correspondientes, se le conoce como modelo de casos de uso.

Las historias de usuario (Cohn, 2004) son especificaciones cortas escritas en una o dos frases utilizando el lenguaje del usuario final. Estas descripciones se complementan por lo habitual tanto con conversaciones que definen el detalle del servicio especificado en la historia de usuario como con descripciones textuales de las pruebas que servirán para determinar si esta fue atendida durante el diseño y la implementación. Debemos destacar que en el título de esta sección usamos la palabra funcional para hacer explícito que las historias de usuario a las que nos referimos describen requerimientos de usuario. Esto es importante porque en algunas ocasiones las historias se emplean para describir otros tipos de requerimientos, por ejemplo, de atributos de calidad.

2.4.3 Documento de especificación de requerimientos

Además del de visión y alcance y del de requerimientos de usuario, otro documento representativo de la ingeniería de requerimientos es el de Especificación de Requerimientos (SRS, por sus siglas en inglés). Como se observa en la figura 2-2, este documento contiene elementos para especificar los atributos de calidad, interfaces externas y restricciones y requerimientos funcionales que, como lo hemos mencionado, representan información relevante a efecto de identificar los *drivers*.

2.5 MÉTODOS PARA LA IDENTIFICACIÓN DE *DRIVERS ARQUITECTÓNICOS*

Hasta el momento, en este capítulo nos hemos limitado a abordar la etapa de requerimientos en términos de descripciones muy generales sobre las tareas de identificación, especificación y priorización de *drivers arquitectónicos*. Sin embargo, poco hemos dicho acerca de cómo llevar a cabo estas actividades. En la actualidad existen algunos métodos y modelos que pueden ser utilizados de una forma más sistemática por el arquitecto en su trabajo en aquella etapa. Algunos de ellos son:

1. Taller de atributos de calidad (QAW, por sus siglas en inglés).
2. Método de diseño centrado en la arquitectura (ACDM, por sus siglas en inglés).
3. Funcionalidad, usabilidad, confiabilidad, desempeño, soporte (FURPS+).

En las siguientes secciones describiremos estos métodos.

2.5.1 Taller de atributos de calidad (QAW)

El taller de atributos de calidad (Barbacci, Ellison, Lattanze, Stafford, Weinstock y Wood, 2008), o QAW[1], es un método desarrollado por el SEI que define un proceso para llevar a cabo de forma sistemática la identificación de *drivers* de atributos de calidad.

El método se basa en realización de un taller, que dura de uno a dos días e incluye diversas actividades para lograr la identificación, especificación y priorización de requerimientos de atributos de calidad. Tiene dos características distintivas: la primera es que considera la participación activa de los diversos tipos de interesados, por ejemplo los dueños, usuarios directos, dueños, patrocinadores, desarrolladores o los que le darán mantenimiento; y la segunda es que utiliza escenarios para la especificación de los *drivers* de atributos de calidad.

El método considera también la participación de un grupo de facilitadores (un líder y uno o más secretarios) con experiencia en este tipo de talleres, quienes apoyan la realización en tiempo y forma de las actividades.

Los escenarios de atributos de calidad son análogos a los casos de uso o historias de usuario funcionales respecto de que especifican requerimientos de atributos de calidad.

[1] En inglés: *Quality Attribute Workshop*.

Aunque en la práctica existen variaciones en relación con los elementos de información que debe contener un escenario de tales atributos, se recomienda que este se elabore en términos de la fuente y el estímulo que produce determinada respuesta del sistema, las partes del sistema afectadas por el estímulo, el contexto de ejecución en el cual se producirá dicha repuesta y, por sobre todo, la medida de calidad asociada a la respuesta producida. La figura 2-6 muestra un extracto de un escenario para el atributo de calidad de desempeño. El escenario corresponde al sistema de venta de boletos de autobús que hemos establecido; en específico, describe el funcionamiento esperado del sistema cuando se lleva a cabo la operación de búsqueda de corridas. En el contexto del caso de estudio de este libro, en la sección 2.2 del apéndice se encuentran, entre otros, ejemplos de escenarios de atributos de calidad, como desempeño y disponibilidad.

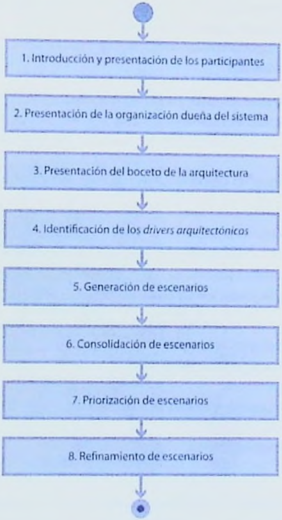
Estímulo:	Petición de búsqueda de corridas de autobús.
Fuente del estímulo:	Un usuario del sistema.
Contexto de operación:	En un momento normal de operación con no más de 99 usuarios conectados al sistema.
Artefacto:	El sistema.
Respuesta:	El sistema procesa la petición y muestra la lista de corridas correspondientes.
Medida de respuesta:	Se procesa la petición en un tiempo no mayor a 15 segundos.

› Figura 2-6 Extracto de un escenario para el atributo de calidad de desempeño

Para llevar a cabo el taller de atributos de calidad es necesario que la organización para la cual se desarrollará el sistema tenga claridad sobre el contexto de operación de este, así como de sus principales funcionalidades, expectativas de calidad y restricciones.

Como se ilustra en la figura 2-7 el taller define un proceso secuencial de ocho etapas, las cuales describimos a continuación.

1. **Introducción y presentación de los participantes.** Uno de los facilitadores describe el objetivo, las fases y las actividades a realizarse en el taller y permite que los demás participantes se presenten.
2. **Presentación de la organización dueña del sistema.** Un representante de la organización para la cual se desarrollará el sistema explica respecto de este, su contexto, los objetivos de sus principales funcionalidades, las expectativas de calidad y las restricciones. Mientras transcurre esta explicación, uno de los facilitadores recaba la información que considera relevante para identificar *drivers arquitectónicos*.
3. **Presentación del boceto de la arquitectura.** Aun cuando no existe todavía una arquitectura definida para el sistema, durante el taller se solicita al arquitecto de *software* de la organización presentar un boceto o primera versión de ella, la cual él considera que satisfará los requerimientos descritos en el paso anterior. De forma similar, durante la presentación uno de los facilitadores registra la información importante para identificar *drivers arquitectónicos*.
4. **Identificación de los *drivers* arquitectónicos.** Con base en la información presentada hasta el momento, los facilitadores definen una primera versión del conjunto de *drivers arquitectónicos* del sistema poniendo énfasis especial en los que corresponden a atributos de calidad. Después los exhiben a los participantes del taller y, mediante una dinámica de lluvia de ideas, los invitan a hacer preguntas



› **Figura 2-7** Etapas del taller de atributos de calidad

y/o comentarios con el propósito de refinarlos. Al término de esta etapa debe haber un consenso sobre la cantidad y el tipo de *drivers* que hayan sido identificados.

5. **Generación de escenarios.** Durante esta etapa los diversos interesados en el sistema generan los escenarios "crudos" o básicos para los atributos de calidad identificados. Esta creación se lleva a cabo mediante una lluvia de ideas en la cual cada participante propone un escenario para un atributo relevante en su contexto. Un facilitador guía esta lluvia de ideas de modo que al final de ella se tenga especificado al menos un escenario para cada uno de los *drivers arquitectónicos* identificados en el paso anterior. Para la especificación de escenarios se utiliza el formato de la figura 2-6.
6. **Consolidación de escenarios.** Uno de los facilitadores pide a los participantes identificar los escenarios que son similares; esto se hace con el propósito de consolidarlos y generar un conjunto reducido, pero completo, de estos. Al término de la etapa debe haber un consenso sobre el número y tipo de los escenarios especificados.
7. **Priorización de escenarios.** Por medio de una dinámica de votación, en esta etapa se lleva a cabo la priorización de los escenarios especificados en el paso anterior. Los facilitadores otorgan a cada participante un número determinado de votos y los invitan a asignarlos a los que consideran de mayor prioridad. De manera ideal, los votos asignados a los participantes individuales corresponde a 30% del número total de escenarios. La votación ocurre en dos rondas, en cada una de las cuales ellos utilizan la mitad de sus votos.

8. **Refinamiento de escenarios.** En esta etapa se especifican con mayor detalle los escenarios de más prioridad, por lo regular cuatro o cinco. Dicho detalle incluye por ejemplo, información sobre los objetivos negocio que soporta el escenario, definiciones de los atributos de calidad o problemas identificados para dichos escenarios.

Al término de la realización de estas etapas se debe tener identificado y/o especificado: un conjunto de *drivers arquitectónicos*, una serie de escenarios correspondientes a los de atributos de calidad (algunos especificados con más detalle que otros) y una lista con los escenarios de atributos de calidad priorizados.

2.5.2 Método de diseño centrado en la arquitectura (ACDM–etapas 1 y 2)

El método de diseño centrado en la arquitectura (Lattanze, 2008), o ACDM[3] por sus siglas en inglés, fue desarrollado por Anthony Lattanze y define un proceso para apoyar la realización de diversas tareas de las etapas del ciclo de desarrollo arquitectónico (no solo de la de requerimientos). La figura 2-8 presenta la estructura y fases definidas por este método, las cuales son las ocho siguientes:

1. Identificación de *drivers arquitectónicos*.
2. Especificación del alcance del proyecto.
3. Creación o refinamiento de la arquitectura.
4. Revisión de la arquitectura.
5. Decisión de llevar o no la arquitectura a producción.
6. Experimentación.
7. Planeación de la implementación.
8. Implementación.

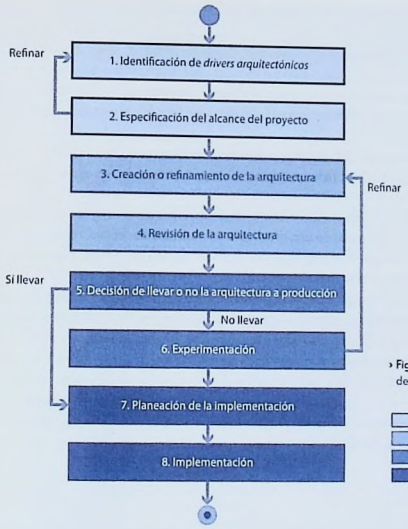
Puesto que estamos describiendo métodos que dan soporte a la etapa de requerimientos, solo abundaremos en las etapas 1 y 2. Durante ellas se llevan a cabo actividades que permiten la identificación, especificación y priorización de *drivers arquitectónicos*.

El propósito de la identificación de *drivers* es reunirse con representantes de la organización que requiere el sistema, a efecto de registrar la mayor cantidad de información relacionada con los *drivers arquitectónicos*. Sin embargo, los *drivers* obtenidos en esta etapa quedarán en forma "cruda" pues no se realiza algún análisis, refinamiento o consolidación exhaustiva de ellos.

Para dar soporte a esta etapa, el método propone definir siete roles en el equipo de desarrollo: administrador del proyecto, ingeniero de soporte técnico, arquitecto de *software* líder, ingeniero de requerimientos, experto en tecnología, ingeniero de calidad e ingeniero de producción. Respecto de los roles en la organización que requiere el sistema, no hay un conjunto definido, y la decisión sobre qué involucrados deben participar en la reunión para identificar los *drivers arquitectónicos* la toma el equipo de desarrollo. Sin embargo, se menciona que podrían tomar en conjunto esta decisión ingenieros y expertos de dominio, desarrolladores, diseñadores, integradores, capacitadores, personal de adquisiciones, usuarios finales, administradores del sistema, directivos de la organización, administradores y personal de ventas.

Las actividades para dar soporte a la identificación ocurren en el marco de un taller de *drivers arquitectónicos*, el cual tiene naturaleza similar al QAW. Dependiendo del tamaño o complejidad del sistema, durante la etapa de identificación de *drivers arquitectónicos* podría realizarse más de un taller. El método provee un proceso definido, así como una serie de técnicas, recomendaciones y formatos que pueden utilizarse para facilitar la planeación y realización del taller.

[3] *Architecture Centric Design Method*



› **Figura 2-8** Etapas del método de diseño centrado en la arquitectura

El proceso necesita que algunos de los miembros del equipo de desarrollo asuman tres roles requeridos en el marco del taller: un facilitador —por lo habitual desempeñado por el ingeniero de requerimientos—, un cronometrador y uno o más secretarios.

El taller se compone por lo habitual de seis fases:

1. **Introducción y presentación de los participantes.** Un representante del equipo de desarrollo de la arquitectura, que la mayoría de las veces es el administrador del proyecto, describe el objetivo, las etapas y las actividades a realizarse en el taller y permite que se presenten los demás participantes.
2. **Descripción del sistema.** Un representante de la organización explica el contexto y los objetivos del sistema. Mientras esto transcurre, los secretarios recaban la información relevante a efecto de identificar los *drivers arquitectónicos*.
3. **Identificación de descripciones operacionales.** Un representante del equipo de desarrollo de la arquitectura lleva a cabo diversas técnicas para propiciar que los representantes de la organización hagan descripciones acerca de las formas en que ellos se imaginan que van a interactuar con el sistema. Los secretarios y el resto del equipo escriben la información recabada en formatos sugeridos por el método y con notas personales, respectivamente.
4. **Identificación de atributos de calidad.** De forma similar al paso anterior, un representante del equipo de desarrollo de la arquitectura lleva a cabo diversas técnicas para propiciar que los representantes de la organización describan los atributos de calidad del sistema. Los secretarios y el resto del equipo

anotan la información recabada en formatos sugeridos por el método y por medio de notas personales, respectivamente.

5. **Identificación de restricciones técnicas y de negocio**. De forma similar al paso anterior, un representante del equipo de desarrollo lleva a cabo diversas técnicas para propiciar que quienes representan a la organización describan las restricciones técnicas y de negocio del sistema. Los secretarios y el resto de los miembros escriben la información recabada en formatos sugeridos por el método y mediante notas personales, respectivamente.
6. **Resumen y acciones siguientes.** El facilitador presenta un resumen de la información recabada e indica a los asistentes que serán contactados posteriormente para obtener mayores detalles sobre esta.

Concluido el taller, el equipo de la *arquitectura de software* se debe reunir para consolidar la información y generar los documentos necesarios, como casos de uso, escenarios de atributos de calidad, entre otros. Además del conjunto de *drivers*, durante esta etapa se crea también una primera versión de un plan maestro de diseño, que se irá actualizando durante las etapas restantes de método.

Por otra parte, el propósito de la etapa de especificación del alcance del proyecto es analizar, consolidar y priorizar los *drivers arquitectónicos* obtenidos en la etapa de identificación de estos. Del análisis se deben generar las especificaciones finales de los *drivers*. Al término, todos ellos deben estar tanto especificados de forma clara y completa como priorizados. Esto hace que sean comprendidos por los interesados en el sistema, incluyendo al equipo de diseño de la arquitectura. Para la especificación de los *drivers* de requerimientos funcionales, el método sugiere echar mano de casos de uso; para la de los de requerimientos de atributos de calidad se utilizan escenarios. De manera similar que para la fase anterior, el método provee plantillas y guías para llevar a cabo el análisis, consolidación y priorización de los *drivers*. Si es necesario, en esta etapa se realizan también actualizaciones al plan maestro de diseño.

2.5.3 FURPS+

Más que un método, FURPS+ es un modelo creado por Hewlett-Packard, el cual define una clasificación de atributos de calidad de *software:* Funcionalidad (*Functionality*), Usabilidad (*Usability*), Confiabilidad (*Reliability*), Desempeño (*Performance*) y Soporte (*Supportability*). El modelo considera también restricciones de diseño, de implementación, físicas y de interfaz. El signo + con el que termina el acrónimo se refiere a todas estas restricciones.

Las restricciones de diseño y de implementación denotan aspectos que restringen el diseño del sistema y su codificación, respectivamente. Las de interfaz describen aspectos relevantes acerca del comportamiento de los elementos externos con los que el sistema debe interactuar. Por último, las restricciones físicas especifican propiedades que debe tener el *hardware* en el que este se instalará.

Por lo tanto, el modelo considera todos los *drivers arquitectónicos*. Es importante mencionar también que el Proceso Unificado de Rational (RUP[4], por sus siglas en inglés), un método para el análisis, diseño, implementación y documentación de sistemas orientados a objetos, toma en cuenta el uso de FURPS+ de forma explícita (Kruchten, 1995).

2.5.4 Comparación de los métodos y el modelo

Con el propósito de proveer al lector una una vista consolidada de las características de los métodos descritos en las secciones anteriores, presentamos la siguiente tabla comparativa.

[4] *Rational Unified Process*.

	Taller de atributos de calidad (QAW)	Método de diseño centrado en la arquitectura (ACDM)	FURPS+
Tipo	Taller	Taller	Modelo
Duración	Uno a dos días, dependiendo de: tamaño del proyecto, estado actual del diseño, lugar de realización del taller y número de participantes.	Uno o más talleres, con duración de uno a dos días, que se llevan a cabo de forma centralizada o distribuida, dependiendo de: tamaño del proyecto, estado actual del diseño, lugar de realización del taller y número de participantes.	Puede ser utilizado dentro del contexto de un método de desarrollo para dar soporte a la especificación y documentación de *drivers* de la arquitectura.
Mecánica y enfoque	Se llevan a cabo presentaciones y lluvia de ideas, con la participación activa de diversos interesados en el sistema.	Se llevan a cabo presentaciones y lluvia de ideas, con la participación activa de diversos interesados en el sistema.	n/a
Provee un proceso que especifica las actividades, funciones y artefactos de entrada y salida.	Sí	Sí	n/a
Participantes	Se definen dos roles principales por parte del equipo de desarrollo: líder y secretario. Se asumen otros por parte de la organización cliente, pero no se especifican.	Se definen cuatro roles principales por parte del equipo de desarrollo: líder del equipo de diseño de la arquitectura, facilitador, cronometrador y secretario, además de la participación de alguno de trece posibles por parte de la organización que requiere el sistema.	n/a
Entradas	Información sobre el contexto de operación del sistema, así como de información general sobre las principales funcionalidades, expectativas de calidad y restricciones de este.	Información sobre el contexto de operación del sistema, así como de información general sobre las principales funcionalidades, expectativas de calidad y restricciones de este.	n/a
Salidas	Un conjunto de *drivers arquitectónicos*, una serie de escenarios de atributos de calidad (algunos especificados con más detalle que otros) y una lista con los escenarios de atributos de calidad priorizados.	Todos los *drivers arquitectónicos* del sistema completamente documentados y priorizados.	n/a
Criterios de terminación	Que se hayan especificado completamente los escenarios de atributos de calidad de mayor prioridad.	Que todos los *drivers arquitectónicos* hayan sido completamente documentados.	n/a
Considera las tres clases de *drivers* principales: requerimientos funcionales, atributos de calidad y restricciones.	No. El énfasis está sobre *drivers* de atributos de calidad.	Sí	Sí