CAPÍTULO 1

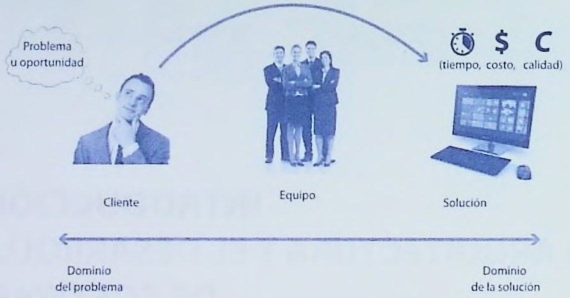
INTRODUCCIÓN: LA ARQUITECTURA Y EL DESARROLLO DE *SOFTWARE*

En la actualidad, el *software* está presente en gran cantidad de objetos que nos rodean: desde los teléfonos y otros dispositivos que llevamos con nosotros de forma casi permanente, hasta los sistemas que controlan las operaciones de organizaciones de toda índole o los que operan las sondas robóticas que exploran otros planetas. Uno de los factores clave del éxito de los sistemas es su buen diseño; de manera particular, el diseño de lo que se conoce como *arquitectura de software*.

Este concepto, al cual está dedicado el presente libro, ha cobrado una importancia cada vez mayor en la última década (Shaw y Clements, 2006). En este capítulo presentamos la introducción al tema y una visión general de la estructura de este libro.

1.1 VISIÓN GENERAL DEL DESARROLLO DE SISTEMAS DE *SOFTWARE*

Expresado de manera simplificada, el desarrollo de un sistema de *software* puede verse como una transformación hacia la solución técnica de determinada problemática u oportunidad con el fin de resolverla, como se muestra en la figura 1-1. Este cambio enfrenta a menudo restricciones en relación con el tiempo, el costo y la calidad.

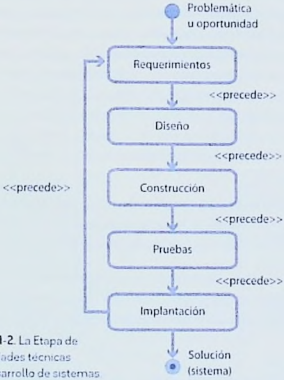


› **Figura 1-1.** Visión simplificada del desarrollo de sistemas

Durante la transformación, que inicia en el dominio del problema y culmina en el de la solución, se llevan a cabo distintas *actividades técnicas,* las cuales se describen enseguida y se muestran en la figura 1-2.

- **Requerimientos.** Se refiere a la identificación de las necesidades de clientes y otros interesados en el sistema, y a la generación de especificaciones con un nivel de detalle suficiente acerca de lo que el sistema debe hacer.
- **Diseño.** En esta etapa se transforman los requerimientos en un diseño o modelo con el cual se construye el sistema. Hace alusión esencialmente a tomar decisiones respecto de la manera en que se resolverán los requerimientos establecidos previamente. El resultado del diseño es la identificación de las partes del sistema que satisfarán esas necesidades y facilitarán que este sea construido de forma simultánea por los individuos que conforman el equipo de desarrollo.
- **Construcción.** Se refiere a la creación del sistema mediante el desarrollo, y prueba individual de las partes que lo componen, para su posterior integración, es decir, conectar entre sí las partes relacionadas. Como parte del desarrollo de las partes, estas se deben diseñar en detalle de forma individual, pero este diseño detallado de las partes es distinto al diseño de la estructuración general del sistema completo descrito en el punto anterior.
- **Pruebas.** Actividad referida a la realización de pruebas sobre el sistema o partes de este a efecto de verificar si se satisfacen los requerimientos previamente establecidos e identificar y corregir fallas.
- **Implantación.** Llevar a cabo una transición del sistema desde el entorno de desarrollo hasta el entorno donde se ejecutará de forma definitiva y será utilizado por los usuarios finales.

Por simplificar, en estas actividades no se considera el mantenimiento, aunque también es muy importante en el desarrollo de sistemas.



› **Figura 1-2.** La Etapa de Actividades técnicas del desarrollo de sistemas.

Hay una relación de precedencia entre las actividades descritas: se precisa hacer por lo menos algo de requerimientos antes de diseñar; un tanto de diseño antes de construir; por lo menos algo de construcción antes de probar, y algunas pruebas antes de implantar. Que estas actividades se hagan por completo o de forma parcial, depende del tipo de ciclo de desarrollo que se elige, el cual va desde lo puramente secuencial (cascada) hasta lo completamente iterativo.

La *arquitectura de software* tiene que ver principalmente con la actividad de diseño del sistema; sin embargo, juega también un rol importante en relación con las demás actividades técnicas, como veremos más adelante.

1.2 DEFINICIÓN DE *ARQUITECTURA DE SOFTWARE*

Al igual que con diversos términos en *ingeniería de software*, no existe una definición universal del concepto de *arquitectura de software*.[1] Sin embargo, la definición general siguiente que propone el Instituto de Ingeniería de *Software* (SEI, por sus siglas en inglés) tiende a ser aceptada ampliamente:

> La ***arquitectura de software*** de un sistema es el conjunto de estructuras necesarias para razonar sobre el sistema. Comprende elementos de ***software***, relaciones entre ellos, y propiedades de ambos.
> (Bass, Clements y Kazman, 2012)

[1] Para muestra basta ver la colección de definiciones que mantiene el SEI en la página web http://www.sei.cmu.edu/architecture/start/glossary/community.cfm

De esta definición, el término "elementos de *software*" es vago, pero una manera de entenderlo es considerar de forma individual las partes del sistema que se deben desarrollar, las cuales se conocen como *módulos*. Por otro lado, las propiedades de estos elementos se refieren a las *interfaces*: los contratos que exhiben estos módulos y que permiten a otros módulos establecer dependencias o, dicho de otro modo, conectarse con ellos. El establecimiento de interfaces bien definidas entre elementos es un aspecto fundamental en la integración y la prueba exitosa de las partes de un sistema desarrolladas por separado, por ello juegan un rol esencial en la arquitectura.

Es importante señalar que el término "elementos" de la definición previa no se refiere únicamente a módulos. El diseño de un sistema requiere que se piense no solo en aspectos relacionados con el desarrollo simultáneo por un grupo de individuos, sino también con la satisfacción de requerimientos, la integración y la implantación. Para ello es necesario considerar tanto el comportamiento del sistema durante su ejecución como el mapeo de los elementos en tiempo de desarrollo y ejecución hacia elementos físicos. Por lo anterior, el término "elementos" puede hacer referencia a:

- Entidades dadas en el tiempo de ejecución, es decir, dinámicas, como objetos e hilos.
- Entidades que se presentan en el tiempo de desarrollo, es decir, lógicas, como clases y módulos.
- Entidades del mundo real, es decir, físicas, como nodos o carpetas.

Al igual que con los módulos, todos estos elementos se relacionan entre sí mediante interfaces u otras propiedades, y al hacerlo dan lugar a distintas *estructuras*. Es por ello que cuando se habla de la arquitectura de un sistema no debe pensarse en solo una estructura, sino considerarse una combinación de estas, ya sean dinámicas, lógicas o físicas.

1.3 ARQUITECTURA, ATRIBUTOS DE CALIDAD Y OBJETIVOS DE NEGOCIO

Además de ayudar a identificar módulos individuales que permitan llevar a cabo el desarrollo en paralelo de un sistema por parte de un equipo de desarrollo, la *arquitectura de software* tiene otra importancia especial: la manera en que se estructura un sistema tiene impacto directo sobre la capacidad de este para satisfacer los requerimientos, en particular aquellos que se conocen como *atributos de calidad* del sistema (de ellos hablaremos en detalle en el capítulo 2).

Ejemplos de estos atributos incluyen el desempeño, el cual tiene que ver con el tiempo de respuesta del sistema a las peticiones que se le hacen; la usabilidad (o facilidad de uso), relacionada con qué tan sencillo es para los usuarios hacer operaciones con el sistema, o bien, la modificabilidad (o facilidad de modificación), la cual tiene que ver con qué tan simple es introducir cambios en el sistema.

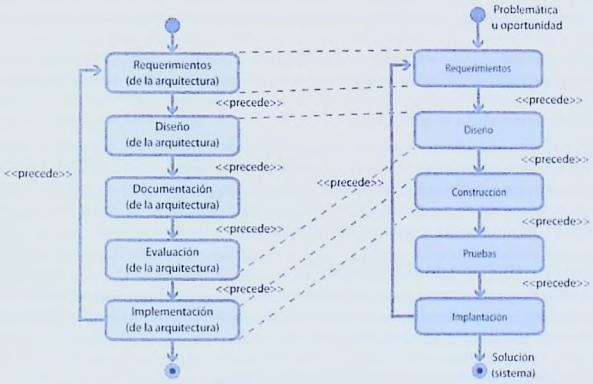
Las *decisiones de diseño* que se toman para estructurar un sistema permitirán o impedirán que se satisfagan los atributos de calidad. Por ejemplo, un sistema estructurado de manera tal que una petición deba transitar por muchos componentes implantados en nodos distintos antes de que se devuelva una respuesta podría tener un desempeño pobre.

De otro lado, un sistema estructurado a modo que los componentes sean altamente dependientes entre ellos (es decir, que estén altamente *acoplados*) limitará severamente la modificabilidad. De manera notable, la estructuración tiene un impacto mucho menor respecto a los requerimientos funcionales. Por ejemplo, un sistema difícil de modificar puede satisfacer plenamente los requerimientos funcionales que se le imponen.

Es de señalar que los atributos de calidad y otros requerimientos del sistema se derivan de lo que se conoce como *objetivos de negocio*. Estos objetivos pertenecen al dominio del problema y son las metas que busca alcanzar una compañía y que motivan el desarrollo de un sistema. Es por ello que algunos autores describen a la arquitectura como un "puente" entre los objetivos de negocio y el sistema en sí mismo. Ejemplos de estos objetivos se aprecian en el documento de visión del caso de estudio presentado en la sección 1 del apéndice del libro.

1.4 CICLO DE DESARROLLO DE LA ARQUITECTURA

De manera similar a lo expuesto en relación con las actividades técnicas para el desarrollo de sistemas, podemos hablar de un *ciclo de desarrollo* de la *arquitectura de software* que engloba actividades particulares. Estas se describen a continuación y se integran a las actividades técnicas del desarrollo de sistemas, como se muestra en la figura 1-3, independientemente de la metodología de desarrollo que se utilice, como veremos en el capítulo 7.



› Figura 1-3. Actividades asociadas al ciclo de desarrollo de la arquitectura (a la izquierda) y su mapeo dentro de las actividades técnicas del desarrollo de sistemas (a la derecha).

1.4.1 Requerimientos de la arquitectura

Esta etapa se enfoca en la captura, documentación y priorización de requerimientos que influyen sobre la arquitectura y que, por lo habitual, se conocen en inglés como *drivers* arquitectónicos.[2] Como se mencionó, los atributos de calidad juegan un rol preponderante respecto de los requerimientos, así que esta etapa hace énfasis en ellos. Otros requerimientos, como los *casos de uso* y *las restricciones*, son también relevantes para la arquitectura. El capítulo 2 se enfoca en esta etapa.

1.4.2 Diseño de la arquitectura

La etapa de diseño es probablemente la más compleja del ciclo de desarrollo de la arquitectura. Durante ella se definen las estructuras de las que se compone la arquitectura mediante la toma de decisiones de diseño. Esta

[2] Por desgracia no hemos encontrado en nuestro idioma una palabra adecuada que tenga pleno significado equivalente a *drivers*; por tal motivo, y aunque esta sea un tanto deficiente, la utilizaremos en el libro.

creación estructural se hace por lo habitual con base en dos clases de soluciones abstractas probadas, llamadas *patrones de diseño* y *tácticas*, al igual que en soluciones concretas como las elecciones tecnológicas, tales como los *frameworks*. El capítulo 3 describe de forma detallada esta etapa.

1.4.3 Documentación de la arquitectura

Una vez que ha sido creado el diseño de la arquitectura, es necesario darlo a conocer a otros interesados en el sistema, como desarrolladores, responsables de implantación, líderes de proyecto o el cliente mismo. La comunicación exitosa depende por lo habitual de que el diseño sea documentado de forma apropiada. A pesar de que durante el diseño se hace una documentación inicial que puede incluir bocetos de las estructuras, o bien capturas de las decisiones de diseño, la documentación formal involucra la representación sus estructuras por medio de *vistas*.

Una vista representa una estructura y contiene por lo habitual un diagrama, además de información adicional que apoya en la comprensión de este. La documentación de la arquitectura es el tema del capítulo 4.

1.4.4 Evaluación de la arquitectura

Dado que la *arquitectura de software* juega un rol crucial en el desarrollo, a efecto de identificar posibles riesgos o problemas es conveniente evaluar el diseño una vez que este ha sido documentado. La ventaja de la evaluación es que representa una actividad que puede realizarse de manera temprana (aun antes de codificar), y que el costo de corrección de los defectos identificados por medio de ella es mucho menor al costo que tendría enmendarlos después de que el sistema ha sido construido. El capítulo 5 trata en detalle este tema.

1.4.5 Implementación de la arquitectura

Una vez establecida la arquitectura, se construye el sistema. Durante esta etapa es importante evitar que ocurran desviaciones respecto del diseño definido por el arquitecto. En el capítulo 6 se habla en detalle acerca de la implementación y su relación con la arquitectura.

1.5 BENEFICIOS DE LA ARQUITECTURA

Se mencionó al principio de este capítulo que el desarrollo de sistemas enfrenta por lo general restricciones en relación con el tiempo, el costo y la calidad. Como se describe a continuación, enfatizar las actividades del ciclo de desarrollo de la *arquitectura de software* aporta diversos beneficios respecto de estas restricciones.

1.5.1 Aumentar la calidad de los sistemas

La relación entre arquitectura y calidad es directa: la arquitectura permite satisfacer los atributos de calidad de un sistema y estos son, a su vez, una de las dos dimensiones principales asociadas con la calidad de los sistemas, siendo la segunda el número de defectos. Hacer una inversión significativa en el diseño arquitectónico contribuye a reducir la cantidad de defectos, la cual, de otra forma, podría traducirse en fallas que impactan negativamente en la calidad.

Un ejemplo de falla asociada con un diseño deficiente ocurre cuando se pone un sistema en producción y se descubre que no da soporte al número de usuarios previsto en un principio.

1.5.2 Mejorar tiempos de entrega de proyectos

La *arquitectura de software* juega un rol importante para que los sistemas sean desarrollados en tiempo y forma. En principio, algunos de los elementos que se identifican dentro de las estructuras arquitectónicas ayudan directamente a llevar a cabo estimaciones más precisas del tiempo requerido para el desarrollo. Por otro lado,

una estructuración adecuada ayuda a asignar el trabajo y facilita el desarrollo en paralelo del sistema por parte de un equipo. Lo anterior optimiza el esfuerzo realizado y reduce el tiempo que toma el desarrollo del sistema.

El diseño de la arquitectura involucra con frecuencia la reutilización, ya sea de soluciones conceptuales o de componentes existentes, y esto ayuda también a reducir de manera significativa el tiempo de desarrollo. Por último, y relacionado con la calidad de manera directa, la reducción de defectos resultante de un buen diseño da como resultado una necesidad menor de volver a realizar el trabajo, lo cual contribuye a que los sistemas se entreguen en los plazos previstos.

1.5.3 Reducir costos de desarrollo

Respecto del costo de un sistema, la arquitectura también es fundamental. La reutilización es un factor importante en el momento de hacer un diseño arquitectónico porque ayuda a reducir costos. Por otra parte, es posible considerar la reutilización como un atributo de calidad del sistema y tomar decisiones de diseño al respecto con la finalidad de lograr una disminución de costos en el desarrollo de sistemas subsecuentes. Reiteramos asimismo que un buen diseño contribuye a aminorar la necesidad de volver a hacer el trabajo y facilita el mantenimiento, lo cual también conduce a bajar los gastos.

1.6 EL ROL DEL ARQUITECTO

Las actividades del ciclo de desarrollo son responsabilidad del rol del *arquitecto de software*, y esta función puede ser cubierta por uno o más individuos (en cuyo caso sería un equipo de arquitectura). Aunque de manera un tanto simplista, este arquitecto debe ser visto como un líder técnico cuya tarea principal es tomar decisiones de diseño pertinentes, a efecto de satisfacer los *drivers* arquitectónicos y demás requerimientos del sistema (debe tener idealmente un buen conocimiento del dominio del problema).

Además, el arquitecto debe comunicar sus decisiones y asegurar que durante la construcción del sistema estas sean respetadas por parte de los miembros del equipo de desarrollo. Los aspectos relacionados con requerimientos, comunicación del diseño y guía al equipo de desarrollo, requieren de una cantidad adecuada de habilidades "suaves" (es decir, no técnicas) incluyendo liderazgo, negociación, y excelente comunicación escrita y oral.

En la actualidad, el rol de arquitecto está presente en diversas compañías de desarrollo de *software*, aunque no forzosamente se tiene tanta claridad respecto de las actividades que ellos deben realizar. Aun con esto, es de destacar que en el año 2015, el sitio de *CNN Money* situaba a la labor de arquitecto de *software* como el primero de los cien mejores trabajos en Estados Unidos[1].

En la actualidad, por desgracia pocos arquitectos que laboran en la industria del desarrollo de *software* han recibido una formación teórica en el tema. Esto se debe a que no es sino hasta épocas recientes que se han establecido de manera más formal los conceptos relacionados con la *arquitectura de software* y que pocas instituciones ofrecen cursos enfocados en el tema. Con frecuencia, el desconocimiento de los principios relativos a este tema impacta de manera negativa en los proyectos de desarrollo.

[1] http://money.cnn.com/gallery/pf/2015/01/27/best-jobs-2015/