Unidad 3 / Escenario 5

Lectura fundamental

Introducción al modelamiento de negocio y requerimientos

Contenido

Palabras clave: análisis de *software*, diseño de *software*, requerimientos, arquitectura, programación orientada a objetos.

1. Modelamiento de negocio

En este flujo, se entiende el negocio para el cual se está desarrollando el sistema y el contexto en el cuál se va a desarrollar el proyecto. En la siguiente tabla se relacionan las principales preguntas que se espera deben resolver en este flujo y las posibles fuentes y herramientas para dar respuesta a dichas preguntas.

Tabla 1. Preguntas a resolver en el modelamiento de negocio vs herramientas

Pregunta	Fuente/Herramienta
¿Cuál es la misión de la empresa?	• Direccionamiento estratégico
¿Cómo ejecuta su misión?	• Modelo de procesos • Diagramas de actividades • Casos de Uso
¿Cuál es la visión de la empresa?	• Direccionamiento estratégico
¿Cómo esperar lograr esa visión?	• Políticas • Objetivos estratégicos, otros
¿Quiénes desarrollan esos procesos?	• Definición y descripción de actores • Modelo de negocio
¿Cómo puede el sistema aportar a la empresa? **¿Cuáles son los motivadores de la empresa para querer en sistema?**	• Justificación
¿Qué riesgos enfrenta el proyecto?	• Identificación de riesgos del proyecto
¿Cuál es el alcance y las restricciones del proyecto?	• Económicos • Tecnológicos • De tiempo, entre otros
¿Cuál es el lenguaje de la organización?	• Glosario

Fuente: elaboración propia

De acuerdo con Hernández (2005), los propósitos que se persiguen al realizarse el modelado del negocio son:

- Entender la estructura y dinámica de la organización.
- Entender los problemas actuales e identificar mejoras potenciales.
- Asegurarse de que los clientes, usuarios finales y desarrolladores tengan una idea común de la organización.
- Derivar los requerimientos del sistema a partir del modelo de negocio que se obtenga.

El modelamiento de negocio busca entender la organización en su situación actual, y también desarrollar la visión de la organización que se pretende alcanzar, es decir, el modelo deseado una vez el nuevo sistema esté implementado, para a partir de esta visión definir los procesos, roles y responsabilidades de esa organización a través de dos modelos fundamentalmente, el modelo de casos de uso del negocio y el modelo de dominio del negocio.

- El modelo de casos de uso del negocio describe los procesos de un negocio y cómo estos interactúan con actores externos, es decir, ¿cuáles son las funciones del negocio? y ¿cómo los usuarios de estas funciones las utilizan?
- El modelo de objetos de negocio ayuda a comprender los conceptos que utilizan los usuarios, los conceptos con los que trabajan y con los que deberá trabajar la aplicación. Es una representación visual de los objetos de negocio y cómo están relacionados. No son las clases del *software* aunque algunas vayan a terminar siéndolo.

La definición del modelo de casos de uso de negocio es necesario identificar los siguientes artefactos.

» **Actor de negocio:** "es cualquier individuo, grupo, entidad, organización, máquina o sistema de información externos; con los que el negocio interactúa." (Hernández, 2005, p. 55). Lo que se modela como actor es el rol que se juega cuando se interactúa con el negocio para beneficiarse de sus resultados.

» **Proceso de negocio:** "es un grupo de tareas relacionadas lógicamente que se llevan a cabo en una determinada secuencia y manera y que emplean los recursos de la organización para dar resultados en apoyo a sus objetivos" (Hernández, 2005, p. 56).

» **Caso de uso del negocio:** representa a un proceso de negocio, por lo que se corresponde con una secuencia de acciones que producen un resultado observable para ciertos actores del negocio. Desde la perspectiva de un actor individual, define un flujo de trabajo completo que produce resultados deseables. (Hernández, 2005, p. 56).

No siempre un negocio está organizado en términos de procesos de negocio, por esto es necesario saber cómo identificar dichos procesos. Para encontrar casos de uso del negocio se pueden clasificar los procesos de negocio en tres categorías.

- **Procesos misionales:** estos son los procesos que generan un valor que es percibido por los clientes, para identificarlos es necesario responder la pregunta ¿cuáles son los servicios que el cliente recibe del negocio?
- **Procesos de soporte o apoyo:** estos procesos no benefician directamente al cliente final, sin embargo, son necesarios para poder ejecutar los procesos misionales. Ejemplo de estos pueden ser: gestión de personal, compras, gestión de tecnología, actividades legales, entre otras.
- **Procesos gerenciales:** estos procesos tienen que ver con la gestión del negocio en su conjunto, ejemplo de estas pueden ser: planeación estratégica, comunicación estratégica, gestión financiera, entre otros.

2. Modelamiento de requerimientos

2.1. ¿Qué son y para qué sirven los requerimientos?

Los requerimientos son las capacidades que debe tener un sistema para la solución de un problema. Deben incluir todos los aspectos necesarios para satisfacer las necesidades de los clientes. Para evaluar la calidad del *software* se verifica si este satisface o no los requerimientos del cliente. `A continuación se explica para qué sirven.

» **Para que los usuarios digan y obtengan lo que quieren**

- La adecuada definición de requerimientos representa la primera herramienta para que la comunicación entre usuarios y desarrolladores ocurra en forma adecuada. Un proceso correcto en este sentido debe facilitar el proceso para las dos partes, asegurando que los usuarios tengan cómo expresar sus necesidades de una manera útil, facilitando el registro de esas necesidades y ofreciendo mecanismos para integrar sus solicitudes en el proceso de construcción de la solución de *software* que les será entregada.

» **Para verificar el diseño**

- Una definición adecuada de requerimientos es un plano, mapa o lista de objetivos contra los cuales es posible y de hecho necesario, verificar el diseño del *software*. Sólo contando con una lista actualizada, completa, clara y real de lo que el cliente necesita, es posible verificar si el diseño propuesto logrará satisfacer al usuario final. Este proceso de verificación debe además hacerse antes de comenzar con la codificación, de tal manera que los cambios que sean necesarios se hagan antes de que se incurran en más gastos y pérdidas de tiempo.

» **Para medir el progreso**

- En muchas organizaciones, la definición de requerimientos, junto con la priorización dada a los mismos, son herramientas para verificar de manera rápida el progreso realizado a lo largo del proyecto. De nuevo, tratándose de una lista completa y real de lo que se debe cumplir cuando se termine el *software*, es supremamente útil como herramienta de contraste para el avance de la implementación.

» **Para entregar y aceptar el producto según criterios precisos**

- Una lista o definición de requerimientos bien hecha se convierte también en una suerte de lista de chequeo mediante la cual se puede constatar que el *software* entregado corresponde con lo requerido por el cliente. En este sentido, un proceso consensual de definición de requerimientos generará uno o varios documentos que podrán ser utilizados en la entrega final al cliente para que este verifique que todo lo que solicitó esté incluido en el producto que se le está entregando. De la misma manera, para el desarrollador es útil tener esa lista, para poder contestar a cualquier solicitud por parte del cliente de una funcionalidad que no fue definida desde el principio del proyecto.

» **Como herramienta orientadora de los procesos de pruebas**

- Si se ha definido de una manera completa y clara la lista de funcionalidades que debe tener la aplicación y los comportamientos que se esperan, los procesos de pruebas cuentan con una base firme para definir los casos de prueba y los resultados idóneos a buscar. Esta es una herramienta invaluable que ahorra tiempo y trabajo al mismo tiempo que ofrece seguridad en un proceso tan importante y delicado como es el proceso de verificación de calidad y pruebas.

» **Para generar confianza**

- De manera final, tener una definición completa y clara de los requerimientos de la aplicación es una herramienta para que los gerentes, los desarrolladores, los analistas, los encargados de pruebas y desde luego los clientes tengan tranquilidad y confianza en el proceso. Partir de la base de que todas las personas involucradas en el proceso están trabajando con miras a cumplir el mismo conjunto de objetivos y funcionalidades, minimizando la probabilidad de desacuerdos y malentendidos posteriores, es una gran ventaja en cualquier proceso de desarrollo de *software*.

2.2. ¿Cuáles son las características deseables de un requerimiento?

Según la IEEE (1998) las características deseables en un requerimiento para que cumpla con su propósito son:

» **Independiente del diseño:** la manera como se escribe un requerimiento no debe depender de decisiones o restricciones en términos de plataformas, lenguajes, sistemas operativos. Los requerimientos deben describir funcionalidades que debe tener la aplicación, sin limitarse por la manera técnica en la que serán implementadas. Dicho de otra manera, dice qué debe hacer el sistema, no cómo será implementado.

» **Priorizable:** esta característica responde a la necesidad de poder ordenar los requerimientos en términos de su importancia dentro del proyecto. Esto se hace con el fin de poder dividir el trabajo más adelante y también con propósitos eventuales de negociación, para poder determinar cuáles son los requerimientos que deben incluirse de manera primordial en una liberación parcial del *software* y a cuáles se debe dedicar más tiempo de investigación, desarrollo y pruebas.

- **Necesario:** cualquier requerimiento que se enuncie durante el proceso de construcción de una aplicación de *software* debe ser necesario para su funcionamiento. Requerimientos que no lo sean solamente consumirán tiempo, recursos y esfuerzo, al mismo tiempo que potencialmente podrían enturbiar el proceso de desarrollo y utilización de la aplicación sin agregar valor a la misma.
- **No ambiguo:** la manera en que se describe un requerimiento no debe dejar lugar a dudas respecto a su intención. Si esto sucede, es posible que haya interpretaciones variadas de lo que se desea con dicho requerimiento, y esto puede producir más adelante dificultades durante el proceso de desarrollo e insatisfacción por parte del cliente.
- **Verificable:** un requerimiento debe estar escrito de tal manera que se pueda comprobar su implementación. La manera como se enuncia debe soportar el hecho de que se verifique más adelante que efectivamente fue implementado y que dicha implementación cumple con lo deseado. Clave en este sentido es definir de manera concreta qué se espera como salida o resultado de la correcta implementación y ejecución del requerimiento.
- **Correcto:** posiblemente sobra decirlo, pero un requerimiento debe ser correcto en su enunciado. Es decir, debe reflejar de manera adecuada lo que se espera de la aplicación y debe contener la descripción de una funcionalidad que sea apropiada para la solución de *software* que se está implementando.
- **Consistente:** en estrecha relación con la característica anterior, un requerimiento debe estar alineado con los demás requerimientos de la aplicación. Un requerimiento no debe nunca contradecir lo que otros digan en términos de la funcionalidad de la aplicación. Detectar y corregir este tipo de inconsistencias es lo que facilita que la aplicación al final de su ciclo de implementación satisfaga de manera cabal la totalidad de las necesidades del cliente.
- **Realizable:** un requerimiento debe ser posible de implementar. A pesar de que en un campo como la tecnología la definición de qué es y no es posible varía con rapidez, debe tratarse de que la definición de los requerimientos de cada aplicación sea, desde el principio, lo suficientemente **aterrizada** como para que se pueda iniciar el desarrollo con una razonable certeza de que se podrá cumplir con todo lo enunciado.

- **Trazable:** un requerimiento es trazable cuando se le puede encadenar con todos los elementos de la definición del *software* que lo generaron y si es el caso, con los elementos que él mismo genera. Esto quiere decir, que debe ser posible identificar para un requerimiento funcional el requerimiento (o requerimientos) de usuario que ayuda a satisfacer y de manera más general aun, el requerimiento de negocio al que contribuye. A futuro dentro del proceso de desarrollo, también debe ser posible hacer este seguimiento hacia abajo en términos de poder identificar las unidades de código y las clases y métodos que contribuyen a la satisfacción del requerimiento enunciado. De esta manera, a lo largo de un proyecto de *software* debería ser posible identificar qué método de qué clase contribuye a la satisfacción de qué requerimiento de negocio, siguiendo todos los pasos intermedios.
- **Conciso:** un requerimiento debe ser escrito de la manera más breve posible, sin que por ello pierda claridad. Esto facilita la lectura y la comprensión de lo que se desea y puede ser de ayuda si se requiere hacer correcciones más adelante.
- **Escrito en forma de debe:** esta es más una recomendación de forma con el fin de alcanzar consistencia en la documentación de los requerimientos. Por lo general se recomienda que sean escritos de esta manera: **la aplicación debe....** así se facilita la lectura de la lista de requerimientos y es posible verificar también la claridad que se tiene al momento de escribirlos. Si un requerimiento no se puede escribir de esta manera, es posible que exista la necesidad de revisar y redefinir qué se espera que describa. Esta es otra herramienta para aclarar los requerimientos de la aplicación.

La idea es generar uno o varios documentos que reflejen la totalidad de lo que el cliente necesita de la aplicación. Estos documentos servirán como punto de partida para que el equipo de análisis y diseño defina la mejor manera de abordar el proceso de construcción de *software* y para que de esta manera se lleve a cabo un proceso ordenado y eficiente desde el principio, orientándose a la satisfacción real de las necesidades del cliente.

Resulta interesante anotar que, si se siguen las recomendaciones presentadas previamente en torno a la estructura y forma de los requerimientos, algunas de esas características permearán la documentación total de dichos requerimientos, es decir, si los requerimientos son correctos de manera individual, el documento de definición de requerimientos lo será también. Si los requerimientos son consistentes entre sí, el documento de definición de requerimientos describirá la funcionalidad del *software* de manera consistente y sin lugar para conflictos. Si se han identificado todos los requerimientos necesarios, la definición total de requerimientos será completa.

En general, una atención adecuada a la definición articular de cada requerimiento y su unificación en uno o varios documentos garantiza la existencia de una base realmente sólida de comprensión y descripción de la aplicación a desarrollar. Esto es fundamental antes de comenzar a escribir una sola línea de código, pues sin una base real y clara, el resultado del proceso de codificación no podrá ser realmente estable.

2.3. Tipos de requerimientos

2.3.1. Requerimientos Funcionales

Los requerimientos funcionales definen qué hace el sistema (a través de la definición de entradas y salidas), es decir, las funciones que dicho sistema debe cumplir. Este tipo de requerimientos son los que definen de una manera detallada las funcionalidades concretas que el *software* debe contener con el fin de permitir a los usuarios cumplir con sus labores, de tal manera que los requerimientos de negocio sean satisfechos. Los requerimientos funcionales de los sistemas también declaran explícitamente lo que el sistema no debe hacer.

2.3.2. Requerimientos NO Funcionales

Son aquellos requerimientos que no se refieren directamente a las funciones específicas que entrega el sistema, sino a sus **propiedades emergentes** como la confiabilidad, disponibilidad, tiempo de respuesta entre otros. Estas propiedades son también conocidas como los atributos de calidad del sistema. De forma alternativa, definen las restricciones del sistema como la capacidad de los dispositivos de entrada/salida y la representación de datos que se utiliza en la interface del sistema.

Atributos de calidad: son características que influyen significativamente en la calidad del *software*. Bass (2013) estableció una clasificación de los atributos de calidad en dos categorías que siguen vigentes hasta ahora:

Observables vía ejecución: son aquellos atributos que se determinan del comportamiento del sistema en tiempo de ejecución, es decir que además son perceptibles por el usuario final.

Tabla 2. Descripción de atributos de calidad observables vía ejecución

Atributo de calidad	Descripción
Disponibilidad (*availability*)	Es la medida de disponibilidad del sistema para el uso
Confidencialidad (*confidentiality*)	Es la ausencia de acceso no autorizado a la información
Funcionalidad (*functionality*)	Habilidad del sistema para realizar el trabajo para el cual fue concebido, es decir, para cumplir los requerimientos funcionales.
Desempeño (*perfomance*)	Es el grado en el cual un sistema o componente cumple con sus funciones designadas, dentro de ciertas restricciones dadas, como velocidad, exactitud o uso de memoria. Según Smith (1993), el desempeño de un sistema se refiere a aspectos específicos o el número de eventos procesados en un intervalo de tiempo. Según Bass (1998) se refiere además a la cantidad de comunicación e interacción existente entre los componentes del sistema.
Confiabilidad (*reliability*)	Es la medida de la habilidad de un sistema a mantenerse operativo a lo largo del tiempo.
Seguridad externa (*safety*)	Ausencia de consecuencias catastróficas en el ambiente. Es la medida de ausencia de errores que generan pérdidas de información.
Seguridad Interna (*security*)	Es la medida de la habilidad del sistema para resistir a intentos de uso no autorizados y negación del servicio, mientras se sirve a usuarios legítimos.

Fuente: elaboración propia basada en Bass (2013)

No observables vía ejecución: son aquellos atributos que se establecen durante el desarrollo o mantenimiento del sistema, por lo tanto, son más perceptibles para los ingenieros, quienes implementan y mantienen el sistema.

Tabla 3. Descripción de atributos de calidad no observables vía ejecución

Atributo de Calidad	Descripción
Configurabilidad (*configurability*)	Posibilidad que se otorga a un usuario experto a realizar ciertos cambios al sistema.
Integrabilidad (*integrability*)	Es la medida en que trabajan correctamente componentes del sistema que fueron desarrollados separadamente para ser integrados.
Integridad (*integrity*)	Es la ausencia de alteraciones inapropiadas de la información.
Interoperabilidad (*Interoperability*)	Es la medida de la habilidad de que un grupo de partes del sistema trabajen con otro sistema. Es un tipo especial de integrabilidad.

Modificabilidad (*modifiability*)	Es la habilidad de realizar cambios futuros al sistema.
Mantenabilidad (*maintainability*)	Es la capacidad de someter a un sistema a reparaciones y evolución. Capacidad de modificar el sistema de manera rápida y a bajo costo.
Portabilidad (*portability*)	Es la habilidad del sistema para ser ejecutado en diferentes ambientes de computación. Estos ambientes pueden ser *hardware*, *software* o una combinación de los dos.
Reusabilidad (*reusability*)	Es la capacidad de diseñar un sistema de forma tal que su estructura o parte de sus componentes puedas de reutilizados en futuras aplicaciones.
Escalabilidad (*scalability*)	Es el grado con el que se pueden ampliar el diseño arquitectónico, de datos o procedimental.
Capacidad de prueba (*testability*)	Es la medida de la facilidad con la que el *software*, al ser sometido a una serie de prueba, puede demostrar sus fallas. Es la probabilidad de que asumiendo que tiene al menos una falla, el *software* fallará en su ejecución de prueba.

Fuente: elaboración propia basada en Bass (2013)

2.4. ¿Qué es la ingeniería de requerimientos?

De acuerdo con Pressman (2010) 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 que se transforman en un sistema funcional. Para lograrlo se requiere desarrollas las siguientes tareas: concepción, indagación, elaboración, negociación, especificación, validación y administración.

2.4.1. Concepción

Esta es la etapa en la que se identifica una necesidad u oportunidad de negocio que puede satisfacerse a través de la construcción o modificación de un producto de *software*, lo que da lugar a concebir el proyecto. En esta etapa se busca entender con claridad cuál es el problema que se requiere resolver a quién le interesa que se de solución a este problema, así como la viabilidad en el desarrollo del proyecto.

2.4.2. Indagación

Esta etapa tiene como propósito de respuesta a las siguientes preguntas: “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.” Pressman (2010, p. 103)

Cristel y Kang (1992) definieron algunos problemas que suelen encontrarse en la etapa de indagación. A continuación, la explicación de cada uno.

Problemas de alcance: el límite del sistema está mal definido, los clientes solicitan más de lo que puede o quiere realizarse o incluyen detalles técnicos que no son necesarios y en lugar de esto pueden confundir más que clarificar los objetivos del sistema. Es común que los clientes hablen de cómo creen que debería darse la solución que de identificar el problema.

Problemas de comprensión: los clientes no están completamente seguros de lo que necesitan, tienen una escaza comprensión de las capacidades y limitaciones de su entorno de computación, no existe un total entendimiento del problema, etc. En algunos casos definen requerimientos contradictorios argumentando “necesidades especiales” no comprendidas.

Problemas de volatilidad: los requisitos cambian con el tiempo. Algunos factores que generan cambios son: legislación, cambios en las políticas o estrategias de negocio, reestructuración, integración con otras organizaciones, etc.

¿Sabía que...?

“Fallamos más a menudo porque resolvemos el problema incorrecto, que porque realizamos una mala solución del problema correcto” (Ackoff, 1974)

2.4.3. Elaboración

Esta etapa se centra en desarrollar modelos refinados de los requerimientos a partir de la información recopilada en las fases de concepción e indagación. “La acción de modelar los requerimientos da como resultado uno o más de los siguientes tipos de modelo.

- Modelos basados en el escenario de los requerimientos desde el punto de vista de distintos actores” del sistema.
- Modelos de datos, que ilustran el dominio de información del problema.
- Modelos orientados a clases, que representan clases orientadas a objetos (atributos y operaciones) y la manera en la que las clases colaboran para cumplir con los requerimientos del sistema.
- Modelos orientados al flujo, que representan los elementos funcionales del sistema y la manera como transforman los datos a medida que se avanza a través del sistema.
- Modelos de comportamiento, que ilustran el modo en el que se comparte el *software* como consecuencia de “eventos” externos.” Pressman (2010)

2.4.4. Negociación

Luego de identificar los requerimientos, es común que se encuentren requerimientos que exceden las posibilidades dados los recursos con los que se cuenta, o que se presenten conflictos con requerimientos que se contradicen. En esta etapa se deben identificar dichos conflictos y priorizar los requerimientos y resolver los conflictos, eliminando, combinando o modificando los requerimientos buscando la mayor satisfacción posible a todos los involucrados.

2.4.5. Especificación

En esta etapa se debe lograr un documento escrito que formalice a través de lenguaje natural y modelos gráficos la descripción de los requerimientos.

2.4.6. Validación

En esta esta se analiza la especificación de los requerimientos para garantizar que cumplen con las características deseadas anteriormente. En esta etapa deben participar tanto los ingenieros que desarrollan la especificación como los usuarios o clientes.

2.4.7. Administración de 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." Pressman (2010).

En conclusión, uno de los factores claves de éxito en el desarrollo de un proyecto que busque satisfacer necesidades a través de la construcción o modificación del *software*, es entender con claridad el contexto del negocio en el que se implementará el *software* y los requerimientos que debe satisfacer y, durante todo el desarrollo del proyecto verificar que todas las acciones y decisiones que se tomen estén orientadas a satisfacer estos requerimientos, los cuales pueden variar en el tiempo.

Referencias

Ackoff, R. (1974). *Redesigning The Future, Wiley*.

Bass, L (2013). *Descripción de Atributos de Calidad no observables vía ejecución*. Recuperado de: http://prof.usb.ve/lmendoza/Documentos/PS-6116/Guia%20Arquitectura%20v.2.pdf

Christel, Michael., & Kang, Kyo. (1992). *Issues in Requirements Elicitation (CMU/SEI-92-TR-012)*. Software Engineering Institute, Carnegie Mellon University Recuperado de: http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=12553

Hernández González, A. (2005). *Identificación de procesos de negocio*. Ingeniería Industrial, XXVI (1), 54-59.

IEEE (1998), *Recommended Practice for Software Requirements Specifications*," in IEEE Std 830-1998, vol., no., pp.1-40

Pressman, R. (2010). *Ingeniería del software. Un enfoque práctico*. México, D. F, México: McGraw Hill

INFORMACIÓN TÉCNICA

FACULTAD DE
INGENIERÍA, DISEÑO
E INNOVACIÓN

Módulo: Ingeniería del *Software* I

Unidad 3: Metodologías de desarrollo de *software*

Escenario 5: Introducción al modelamiento de negocio y requerimientos

Autor: Diana Angélica Cruz Ortega

Asesor Pedagógico: Luisa Esperanza Rincón Jiménez

Diseñador Gráfico: Cristian Navarro

Asistente: Ginna Quiroga