Universidad Continental

Análisis y requerimientos de software

Sandra Wong Durand

INNOVADOR · INTERACTIVO · INTEGRAL · 3i

Datos de catalogación bibliográfica

WONG DURAND, Sandra
Análisis y requerimientos de software: manual autoformativo interactivo / Mg. Sandra Wong Durand. -- Huancayo: Universidad Continental, 2017

Datos de catalogación del Cendoc

Administración Financiera . Manual Autoformativo Interactivo
Sandra Wong Durand

Primera edición digital

Huancayo, octubre de 2017

Versión e-book
Disponible en http://repositorio.continental.edu.pe/
ISBN electrónico N.° 978-612-4196-

Dirección: Emma Barrios Ipenza
Edición: Miguel Ángel Córdova Solís
Miriam Ponce Gonzáles
Asistente de edición: Paúl Juan Gómez Herrera
Asesor didáctico: Susana Beatriz Diaz Delgado
Corrección de textos: Juan Guillermo Gensollen Sorados
Diseño y diagramación: Alexander Frank Vivanco Matos

ÍNDICE

INTRODUCCIÓN

Esta asignatura pertenece a la especialidad de la carrera de Ingeniería de Sistemas e Informática de la modalidad a distancia, es de carácter teórico-práctico y está dirigida a los estudiantes de pregrado. Tiene como propósito desarrollar en el estudiante la capacidad de interpretar los diferentes tipos de modelos de diagramas de software, las perspectivas de modelado y los requerimientos del sistema para diseñar las especificaciones a alto nivel.

En general, los contenidos propuestos en el manual autoformativo se dividen en cuatro unidades: En la Unidad I se hará una introducción a los Fundamentos del Modelado, tipos de modelos y fundamentos de análisis; luego, en la Unidad II, se explicarán los fundamentos y el origen de los requerimientos, además de las técnicas de elicitación de estos; en la Unidad III, abordaremos los conceptos sobre la documentación de los requerimientos y técnicas de especificación de requerimientos; en la Unidad IV se desarrollarán los conceptos de revisiones e inspecciones, prototipos para validación de requerimientos, pruebas de aceptación, validación de atributos, y análisis de requerimientos.

Es recomendable que haga una permanente lectura de estudio de los contenidos desarrollados y de los textos seleccionados que amplían o profundizan el tratamiento de la información, junto con la elaboración de resúmenes y una minuciosa investigación vía internet. El desarrollo del manual se complementa con autoevaluaciones, que son una preparación para la prueba final de la asignatura.

Organice su tiempo para que obtenga buenos resultados. La clave está en encontrar el equilibrio entre sus actividades personales y las actividades que asuma como estudiante. El estudio a distancia requiere constancia; por ello, es necesario encontrar la motivación que lo impulse a hacerlo mejor cada día.

El autor

ORGANIZACIÓN DE LA ASIGNATURA

Resultado de aprendizaje de la asignatura

Al término de la asignatura, el estudiante será capaz de validar la especificación de requerimientos de software para un proyecto de software.

Unidades didácticas

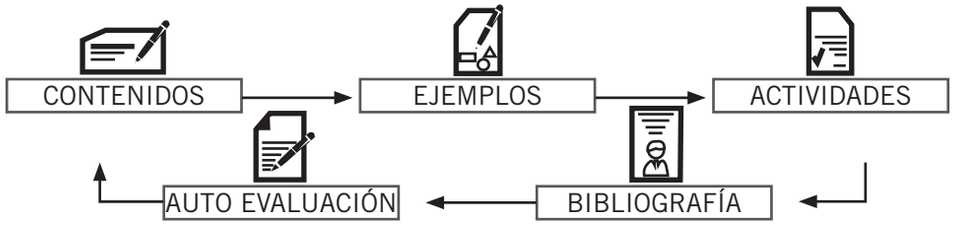
UNIDAD I	UNIDAD II	UNIDAD III	UNIDAD IV
Modelamiento y análisis de software	Elicitación de requerimientos	Especificación de requerimientos de software	Validación de requerimientos
Resultado de aprendizaje	**Resultado de aprendizaje**	**Resultado de aprendizaje**	**Resultado de aprendizaje**
Al finalizar la unidad, el estudiante será capaz de construir el modelamiento de software orientado a objetos de su proyecto.	Al finalizar la unidad, el estudiante será capaz de elicitar los requerimientos de software para su proyecto.	Al finalizar la unidad, el estudiante será capaz de organizar la especificación de requerimientos de software de su proyecto.	Al finalizar la unidad, el estudiante será capaz de validar la especificación de requerimientos de software de su proyecto.

Tiempo mínimo de estudio

UNIDAD I	UNIDAD II	UNIDAD III	UNIDAD IV
Semana 1 y 2	Semana 3 y 4	Semana 5 y 6	Semana 7 y 8
16 horas	16 horas	16 horas	16 horas

UNIDAD I
MODELAMIENTO Y ANÁLISIS DE SOFTWARE

DIAGRAMA DE PRESENTACIÓN DE LA UNIDAD I



ORGANIZACIÓN DE LOS APRENDIZAJES

RESULTADO DE APRENDIZAJE: Al finalizar la unidad, el estudiante será capaz de construir el modelamiento de software orientado a objetos a su proyecto.

CONOCIMIENTOS	HABILIDADES	ACTITUDES
Tema n.° 1: Fundamentos de modelamiento 1. Modelo de sistemas 2. Ciclo de vida del producto 2.1 Predictivo 2.2 Iterativo e incremental 2.3 Adaptativo **Tema n.° 2: Tipos de modelos de sistemas** 1. Modelo de contexto 2. Modelado para el desarrollo estructurado 2.1 Diagrama de flujo de datos 2.2 Modelos de datos 2.2.1. Modelo entidad-relación 2.2.2. Modelo relacional 2.3 Diagrama de transición de estados 2.4 Diccionario de datos 3. Modelado para el desarrollo orientado a objetos 3.1 Rational Unified Process (RUP) 3.2 Lenguaje Unificado de Modelado (UML) 3.2.1 Diagrama de clases 3.2.2 Diagrama de casos de uso 3.2.3 Diagramas de secuencia **Lectura seleccionada n.° 1** *SWEBOK Guide* (2004, pp. 158-165) **Tema n.° 3: Fundamentos del análisis** 1. Modelado basado en escenarios. 2. Modelado orientado al flujo 3. Modelado basado en clases **Lectura seleccionada n.° 2** *Software Requirements* (Wiegers & Beatty, 2013) **Autoevaluación de la Unidad I**	1. Construye modelos de software a partir de una situación real analizada. **Actividad n.° 1** Los estudiantes participan en el foro de discusión sobre modelos de sistemas y modelos de análisis. **Tarea académica n.° 1**	1. Cooperar en la consolidación de los modelos de análisis de software.

Fundamentos de modelamiento

Tema n.° 1

En el siguiente capítulo usted comprenderá los diferentes conceptos de modelamiento y análisis de software asociado a la Ingeniería de Software; además, será capaz de describir los diferentes modelamientos aplicados al desarrollo del software en la fase inicial del ciclo de vida.

1. Modelo de sistemas

Edward Yourdon (1989) en su libro *Análisis Estructurado Moderno* precisa:

> Podemos construir modelos de manera tal que enfatizamos ciertas propiedades críticas del sistema, mientras que simultáneamente desacentuamos otros de sus aspectos. Esto nos permite comunicarnos con el usuario de una manera enfocada, sin distraernos con asuntos y características ajenas al sistema. Y si nos damos cuenta de que nuestra comprensión de los requerimientos del usuario no fue la correcta (o de que el usuario cambió de parecer acerca de sus requerimientos), podemos hacer cambios en el modelo o desecharlo y hacer uno nuevo, de ser necesario. La alternativa es tener algunas reuniones preliminares con el usuario y luego construir todo el sistema; desde luego, existe el riesgo de que el producto final sea aceptable, y pudiera ser excepcionalmente costoso hacer un cambio a esas alturas.
>
> Por esta razón, el analista hace uso de herramientas de modelado para:
>
> - Concentrarse en las propiedades importantes del sistema y al mismo tiempo restar atención a otras menos importantes.
> - Discutir cambios y correcciones de los requerimientos del usuario, a bajo costo y con el riesgo mínimo.
> - Verificar que el analista comprenda correctamente el ambiente del usuario y que lo haya respaldado con información documental para que los diseñadores de sistemas y los programadores puedan construir el sistema (p. 73).

El modelado de sistemas es un componente esencial del desarrollo de software y constituye la base del ciclo de vida. El modelamiento de sistemas empieza con una visión global y luego del análisis inicial se descompone detalladamente creando un proceso o mediante un flujo con entradas y salidas, componentes, supuestos, restricciones, entre otros.

2. Ciclo de vida del producto

El ciclo de vida del producto de software puede ser:

2.1. Predictivo

Este modelo hace referencia a la ejecución de actividades secuenciales del ciclo de vida del desarrollo de software. En este modelo se opta por las validaciones y/o aprobaciones previas de los entregables; por tanto, la fase siguiente no inicia si la anterior no ha sido validada y/o aprobada, pues cada fase requiere información de la etapa previa para iniciar. La desventaja de este modelo es su inflexibilidad, pues las fases del ciclo de vida no pueden ejecutarse en paralelo. Asimismo, cuando en las etapas intermedias se identifican cambios, es necesario volver a la etapa inicial para documentarlos, paralizando las atenciones hasta que se completen las fases previas; como consecuencia, esto puede implicar sobrecostos y desviaciones de plazos. A continuación se muestra el modelo gráficamente:

Figura 1. Modelo predictivo. Fuente: Wong, 2017.

2.2. Iterativo e incremental

Este modelo hace referencia a la ejecución de actividades del desarrollo de software de forma modular; esto quiere decir que se va a desarrollar el software por módulos, cada uno con su ciclo completo. La ventaja de este modelo son las siguientes:

- El usuario final obtiene resultados parciales de su producto.
- Los riesgos de integración del producto se mitigan en etapas tempranas porque la integración se va dando de forma progresiva conforme se va culminando cada módulo.
- El producto de software se puede ir afinando y mejorando en cada iteración.
- Se puede optar por la reutilización de componentes en cada iteración y mejorarlos.

Las desventajas de este modelo son las siguientes:

- Al no priorizarse los requisitos de manera integral desde el inicio, pueden surgir problemas posteriores en la arquitectura.
- El usuario piensa que el producto ya se encuentra terminado en las primeras entregas.

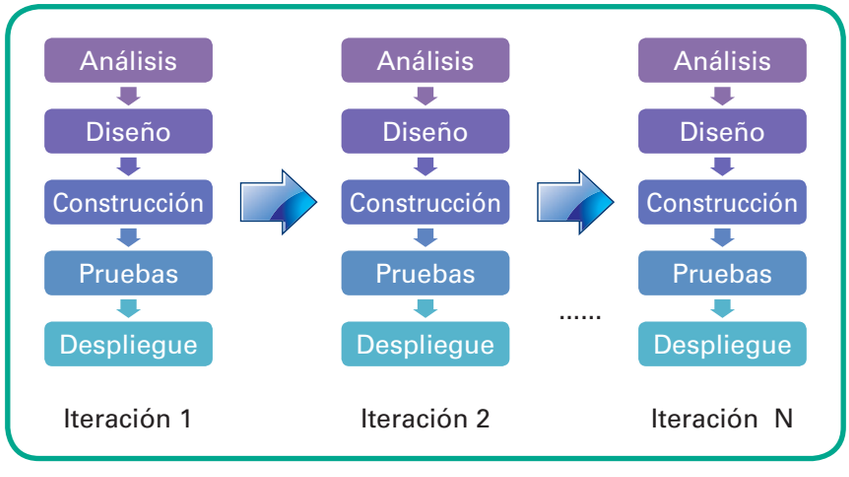


Figura 2. Modelo iterativo e incremental. Fuente: Wong, 2017.

2.3. Adaptativo

Este modelo refiere a la implementación de desarrollo de software con el uso de métodos ágiles; similar a otros modelos, su proceso es cíclico y asume que cada iteración tendrá oportunidades de mejora. Las actividades del desarrollo del software generan funcionalidades del negocio y una

funcionalidad puede tener varios casos de uso en su desarrollo, aquí desaparece el concepto de módulo y existe una retroalimentación continua del desarrollo. Entre los modelos más conocidos de este ciclo de vida tenemos a SCRUM y Extreme programming (XP). Las ventajas de este modelo son las siguientes:

- Es un desarrollo iterativo.
- El desarrollo se centra en los componentes del software.
- Es tolerante a los cambios.
- En cada iteración se revisan las oportunidades de mejora y se aplican los cambios.
- Enfocado en la organización y colaboración de equipo.

Este Modelo DAS se basa en tres fases que son las siguientes:

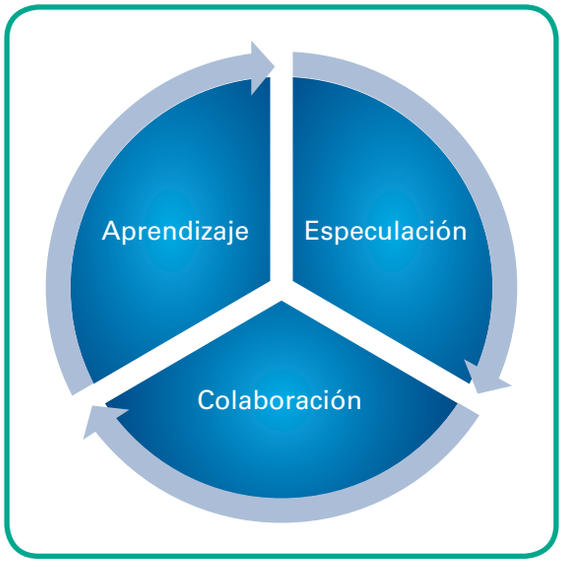


Figura 3: Fases de DAS. Fuente: Wong, 2017.

- **Especulación**: en esta fase se efectúa la planificación preliminar de las entregas que se irán dando como parte del desarrollo del software. A esta fase se le precisa con el término especulación porque se quiere precisar lo impredecible del desarrollo de los sistemas. Si bien es cierto se efectúa una planificación, la misma en cada iteración será susceptible a cambios; la idea es fijar un rumbo que deben seguir el equipo del desarrollo, se debe consignar que el aprendizaje de cada iteración generará mayores ventajas para solucionar futuros inconvenientes con mayor rapidez.

- **Colaboración**: esta fase se centra en el desarrollo del componente, el mismo que refiere a una o muchas funcionalidades por ser desarrolladas como parte de cada iteración. El enfoque del equipo de desarrollo será concluir el componente comprometido para la iteración; el modelo no precisa una metodología o estándar a seguir para el desarrollo, deja libre la posibilidad de la experiencia del equipo de desarrollo para aplicar sus mejores prácticas, colaborar entre sí, apoyarse mutuamente y sacar el producto.

- **Aprendizaje**: esta fase se centra en lo siguiente:

  - La calidad del producto desde la perspectiva del cliente. Mediante técnicas de grupos focalizados el equipo de desarrollo podrá conocer el punto de vista del cliente en relación con el producto desarrollado; aquí se anotarán mejoras que se pueden aplicar al producto.

- La calidad del producto desde la perspectiva técnica. Aquí se efectuará una revisión de diseño, código, base de datos, arquitectura, uso de servicios, entre otros. La revisión se centra en buscar defectos para mejorar y aprender de ellos, no se buscan culpables, la visión del equipo se centra en resolver los problemas, aprender de lo acontecido y proponer mejoras para ser aplicadas en la siguiente iteración; con ello el equipo enriquece sus mejores prácticas.

- El funcionamiento del equipo de desarrollo y las prácticas utilizadas. Uno de los puntos más críticos es el recurso humano; por ello, en este modelo se asigna un tiempo para la revisión del funcionamiento de equipo. Aquí se vislumbra el grado de cohesión, integración y colaboración, el objetivo de equipo debe ser único, no deben primar intereses personales, cada uno debe de cumplir su rol y aportar. El equipo debe discutir de aquello que funcionó bien y de aquello que no, y proponer mejoras para la interacción del equipo.

- Estatus del proyecto, se verifica de manera integral cómo va el proyecto, qué se puede mejorar y ajustar para efectos de la planificación.

El modelo adaptativo es similar al modelo incremental pero aplica prácticas adicionales de gestión en manejo de equipos, así como gestión de usuarios; asimismo se enfoca en el desarrollo de componentes.

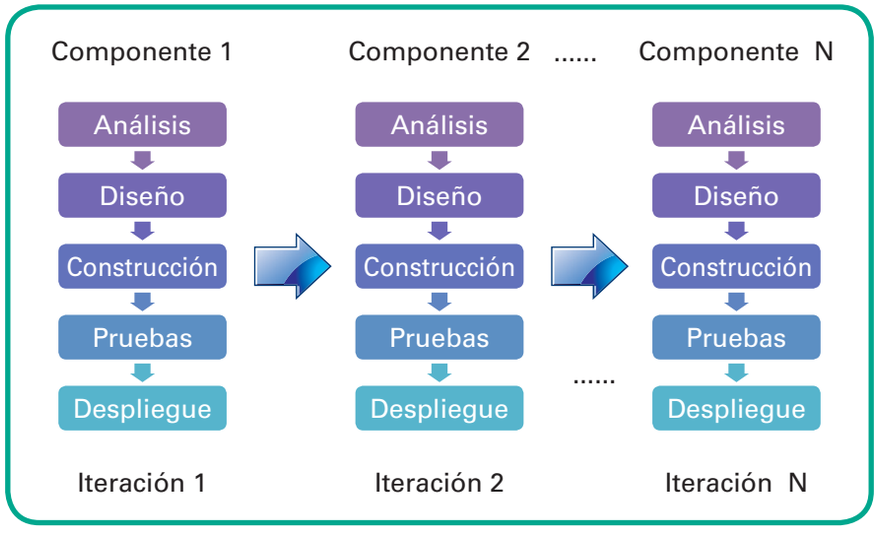


Figura 4: Modelo adaptativo. Fuente: Wong, 2017.

Tipos de modelos

Tema n.° 2

Los modelos de sistemas son representaciones del modelo de negocio de la concepción del requerimiento de software. Estos modelos representarán de forma gráfica el proceso de negocio, lo cual servirá de base para la especificación técnica y, posteriormente, para el desarrollo del software. Los modelos que detallaremos a continuación son representaciones del modelo funcional y de datos de los sistemas; dependiendo del ciclo de vida o metodología por elegir para el desarrollo, se deberá definir el modelo más apropiado para utilizar.

1. Modelo de contexto

Sommerville (2005) en su libro *Software Engineering* menciona los siguientes tipos de modelos:

> En una de la las primeras etapas de la obtención de requerimientos se deben definir los límites del sistema. Esto comprende trabajar conjuntamente con los *stakeholders* del sistema para distinguir lo que es el sistema y lo que el entorno del sistema (p. 155).

Los modelos de contexto son utilizados para las etapas tempranas del relevamiento de información, en los modelos de proceso de negocio; esto, para conocer la funcionalidad y la descripción del requerimiento de usuario a alto nivel.

Este primer diagrama permite validar el primer requisito funcional con el usuario y tener una visión global del sistema. Los requisitos funcionales se irán incrementando conforme se va explotando o detallando el diagrama de contexto; para ello se puede hacer uso de otros diagramas del modelo estructurado o del modelo orientado a objetos.

Por ejemplo, tenemos el desarrollo de un sistema de notificaciones web cuya finalidad es optimizar las comunicaciones formales e informales con el usuario final; lo que el usuario requiere es optimizar los tiempos de generación de la comunicación que se viene trabajando de manera manual y mediante mensajería física. Asimismo, tener conocimiento de que el mensaje ha sido leído por el usuario final.

La solución propuesta implica el desarrollo de un aplicativo web y móvil, para lo cual se deberá implementar lo siguiente:

- Desarrollar el mantenimiento del proceso de Notificación, que actualmente es manual; aquí se registrará el código del proceso de Notificación, por cada tipo de documento que se requiere enviar (carta, oficio, memorándum), se debe vincular el canal de comunicación (Correo, SMS, buzón electrónico) y plantilla que se requiera.

- Desarrollar el mantenimiento de las plantillas por canal (Correo, SMS, buzón electrónico). Este registro permite crear la plantilla y vincularla al canal y al tipo de documento requerido. Será un mantenimiento que podrá ser generado por el usuario final en el momento que se requiera.

- Desarrollar el mantenimiento de responsables. Aquí se anotará a los colaboradores que registrarán el mantenimiento de los documentos de Notificación.

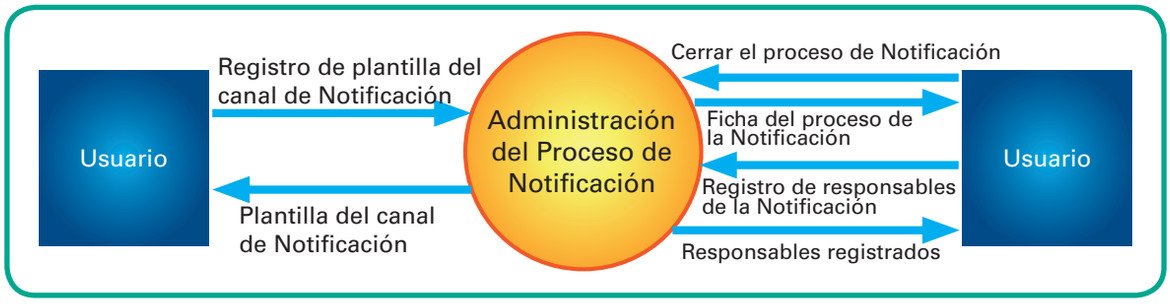


Figura 5: Sistema de notificaciones. Fuente: Wong, 2017.

El diagrama de la figura 5 especifica a alto nivel el requerimiento de usuario descrito previamente. Para efectos del desarrollo de software, este diagrama será la base inicial, y según este se podrán efectuar otros diagramas a mayor detalle.

2. Modelado para el desarrollo estructurado

Se enfoca en la descomposición funcional del sistema, su objetivo es obtener una definición completa del software por desarrollar a través de la elicitación de requisitos. Para efectos del desarrollo, esta definición se descompone en funciones que para este modelo es la unidad básica de la construcción. Asimismo, se definen los datos de entrada y salida. Los diagramas utilizados en este modelo son:

- Diagrama de flujo de datos
- Diagrama entidad–relación
- Diagrama de transición de estados

Cada uno de los diagramas establece una función dentro del modelado del sistema: el diagrama de flujo se centra en la funcionalidad del sistema, el diagrama entidad-relación se enfoca en los datos y el diagrama de transición de estados muestra el comportamiento con base en el tiempo. El diccionario de datos es un diagrama complementario que permite conocer el detalle de los datos.

Figura 6: Componentes de una herramienta CASE para el soporte de métodos estructurados. Fuente: Sommerville, 2005.

2.1. Diagrama de flujo de datos

El diagrama de flujo de datos muestra una perspectiva funcional de un proceso, es útil en el análisis de los requerimientos porque muestran el proceso de negocio de inicio a fin. Si el proceso global contiene subprocesos, el modelo de flujo de procesos se descompone en subprocesos, con el objetivo de otorgar al usuario una visión más detallada del negocio.

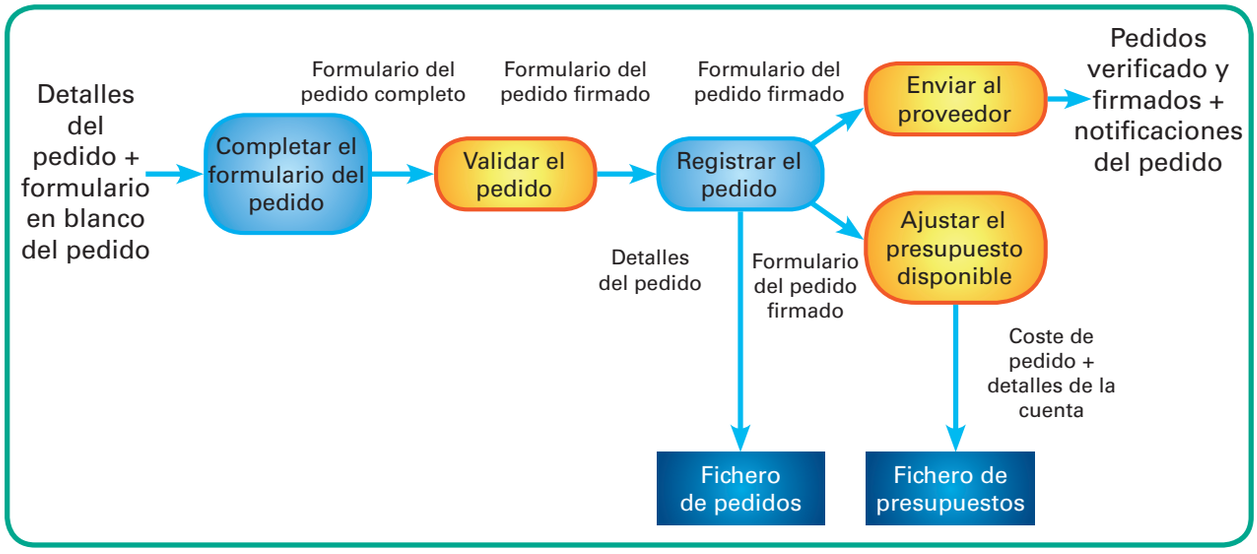


Figura 7: Flujo de datos del procesamiento de un pedido. Fuente: Sommerville, 2005

El modelo de flujo de datos puede ser diagramado en visio o utilizar otras herramientas como SmartDraw o LucidChart, que poseen plantillas predefinidas que ayudan en la elaboración de los diagramas. La simbología que se utiliza para la elaboración de estos diagramas es la siguiente:

Tabla 1
Simbología de un Diagrama de Flujo

Representación gráfica	Descripción
Inicio / Fin	Inicio o fin del programa
Proceso	Descripción del proceso
Entrada	Operaciones de entrada y salida
Decisión	Decisión SÍ/NO
	Conector

Nota: Tomado de Wong, 2017.

2.2. Modelos de datos

Los desarrollos de software incluyen como parte de su implementación la base de datos que contendrá la información relevante del usuario en un repositorio centralizado digital. Muchas veces los sistemas se integran con otros y los datos se pueden grabar en bases de datos ya existentes o también el nuevo sistema puede consumir datos de la base de datos de otro sistema.

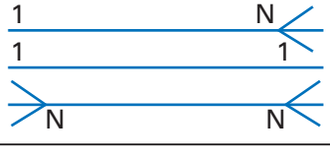
Los diagramas de datos más usados son el diagrama relacional y el diagrama entidad-relación; ambos representan de manera gráfica la descripción de los datos de la aplicación. Cabe mencionar que el diagrama entidad-relación es el utilizado en el modelado estructurado.

2.2.1. Modelo entidad-relación

Describe las necesidades de información del proceso de negocio del usuario mediante un conjunto de entidades y relaciones entre ellas. Asimismo, detalla las descripciones de los datos y restricciones que los datos deben cumplir. Adicionalmente, el modelo contempla los atributos que son las características de la entidad y que para efectos de un modelo físico representarán los datos de una tabla de base de datos. También se describe la cardinalidad, que indica la cantidad de veces que se ejecuta una relación entre las entidades, las cuales pueden ser de uno a uno, de uno a muchos y de muchos a muchos.

A continuación, se muestra la simbología del diagrama:

Tabla 2
Simbología Modelo entidad-relación

Representación gráfica	Descripción
	Entidad
	Atributo
	Cardinalidad
	Decisión SÍ/NO

Nota: Tomado de Wong, 2017.

Las entidades son representaciones del negocio, pueden ser una persona, objeto o animal del cual se describe una característica o acción dentro del software. Por ejemplo a continuación se citan algunas entidades:

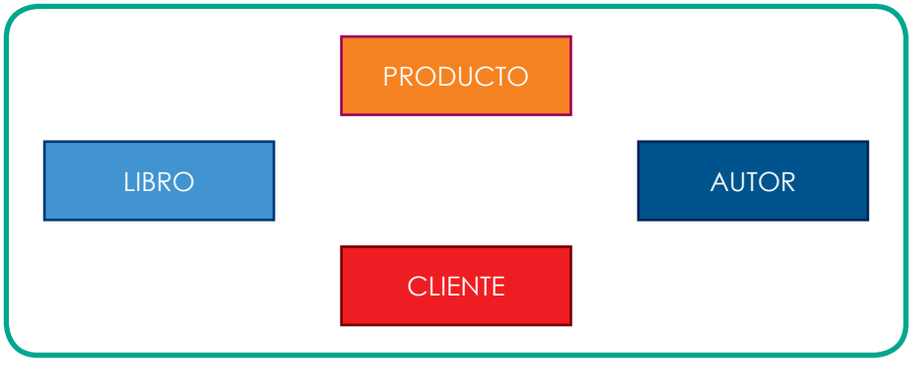


Figura 8: Ejemplo de entidades. Fuente: Wong, 2017.

En la figura 8 se tienen cuatro ejemplos de entidades: la entidad **producto**, la entidad **libro**, la entidad **autor** y la entidad **cliente**; cada una de ellas puede representar en la base de datos una tabla, ya que poseen atributos diferentes; dependiendo del contexto del negocio y la necesidad del usuario se deberán asignar los atributos a cada entidad. Es importante validar con el usuario previamente la necesidad de información que requerirá almacenar en la base de datos del sistema, a fin de no consignar entidades o atributos no requeridos que finalmente nunca contendrán datos.

A continuación, se muestra un ejemplo de entidad con sus atributos:

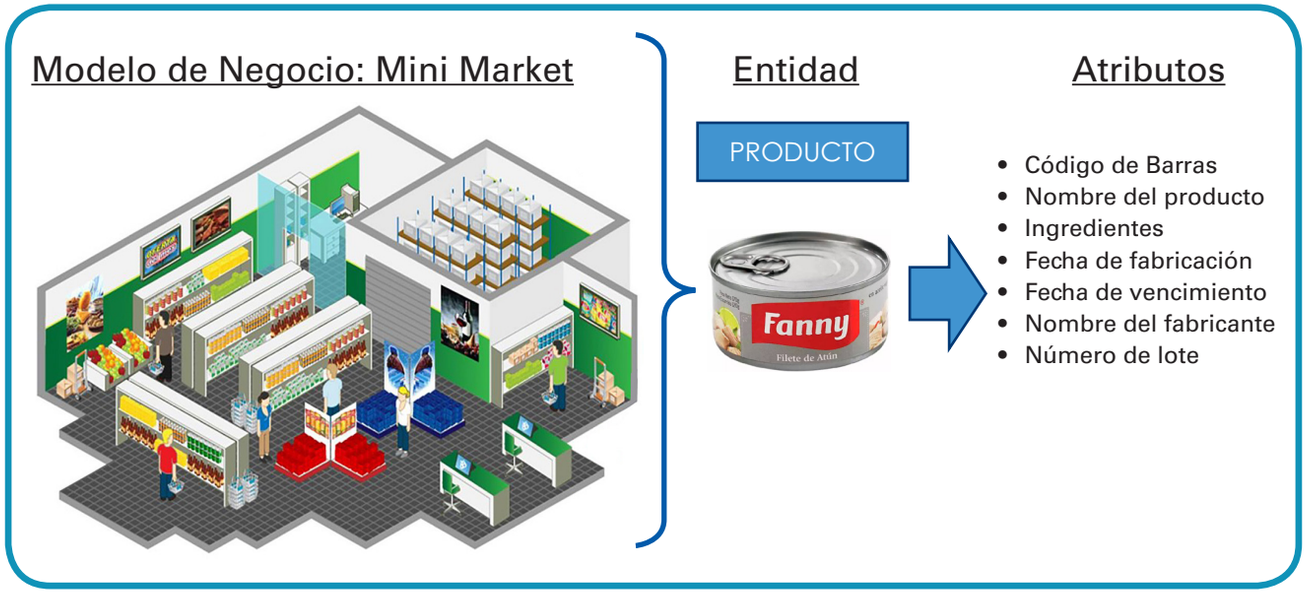


Figura 9: Ejemplo de atributos. Fuente: Wong, 2017.

La figura 9 muestra el modelo de negocio de un minimarket, el mismo que requiere de una entidad "producto" como parte de su sistema. A continuación se describen las características que necesita el producto, tales como código de barras, nombre del producto, ingredientes, fecha de fabricación, fecha de vencimiento, nombre del fabricante y número de lote. Las características de la entidad son consideradas atributos. Entonces, siempre se deberá partir del negocio para primero seleccionar entidades y luego describir sus atributos.
Por ejemplo, a continuación se muestra la notación gráfica del modelo entidad–relación para la entidad persona:

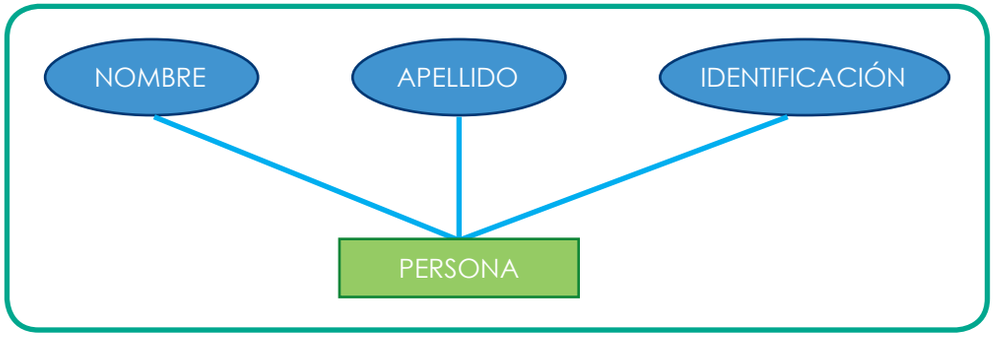


Figura 10: Ejemplo de entidad y atributos. Fuente: Wong, 2017.

Las relaciones son el enlace entre las entidades y describen la acción que se da entre ellas. La siguiente figura muestra la relación entre la persona y el producto mediante la acción comprar; entonces, se tiene que la persona compra un producto y un producto puede ser comprado por una persona.

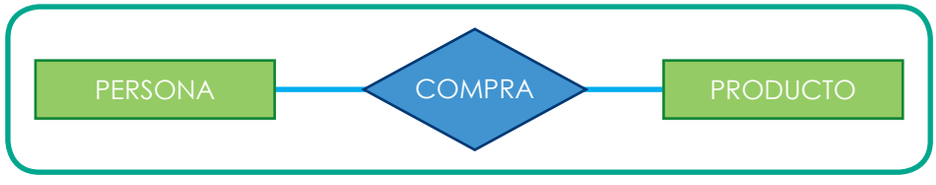


Figura 11: Ejemplo de relación entre entidades. Fuente: Wong, 2017.

Las cardinalidades indican la cantidad de veces máxima y mínima en que puede ocurrir una relación entre las entidades. Así, se tiene lo siguiente:

- Uno a uno: a cada ocurrencia en A le corresponderá una ocurrencia en B, a cada ocurrencia en B le corresponderá una ocurrencia en A. Para el siguiente ejemplo, la lectura será la siguiente: Cada oficina es dirigida por un jefe, un jefe dirige una oficina.

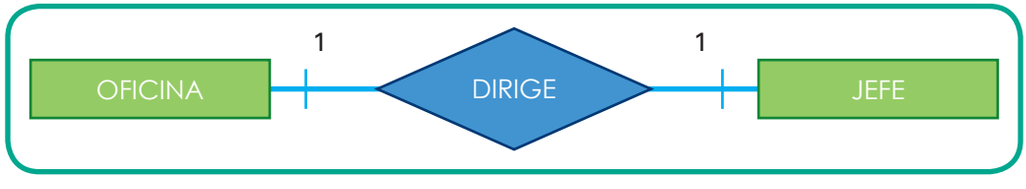


Figura 12: Ejemplo de relación de uno a uno. Fuente: Wong, 2017.

- Uno a muchos: a cada ocurrencia en A le corresponde una o más ocurrencias en B, a cada ocurrencia en B le corresponde una ocurrencia en A. Para el siguiente ejemplo, la lectura será la siguiente: En cada oficina trabajan varios colaboradores, varios colaboradores trabajan en una oficina.

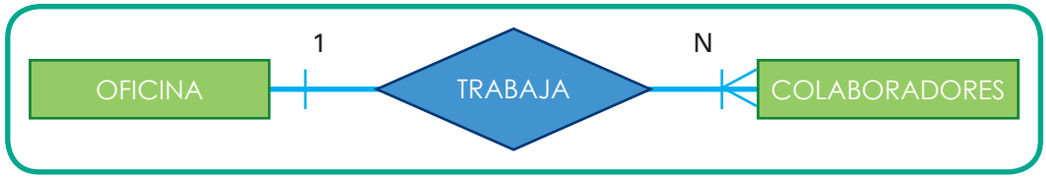


Figura 13: Ejemplo de relación de uno a muchos. Fuente: Wong, 2017.

- Muchos a muchos: a cada ocurrencia en A le corresponden muchas ocurrencias en B y viceversa. El siguiente gráfico muestra la relación de muchos a muchos y se lee de la siguiente manera: Muchos alumnos estudian en muchos cursos, en muchos cursos pueden estudiar muchos alumnos.

Figura 14: Ejemplo de relación de muchos a muchos. Fuente: Wong, 2017.

2.2.2. Modelo relacional

Este modelo, al igual que el modelo entidad-relación, recoge la información de datos que el usuario requiere que se almacene en una base de datos. A diferencia del modelo entidad-relación, el modelo relacional utiliza tablas, campos y relaciones, un modelo similar al modelo físico de base de datos. En este modelo, las tablas son los elementos de almacenamiento principales y se componen de filas (registros) y columnas (campos). Cada tabla tiene una clave primaria, que es el identificador de la tabla. El establecimiento de relación entre las tablas se da mediante la inclusión en forma de columna de la clave primaria de la tabla A en la tabla B; a esta columna se le denomina clave foránea.

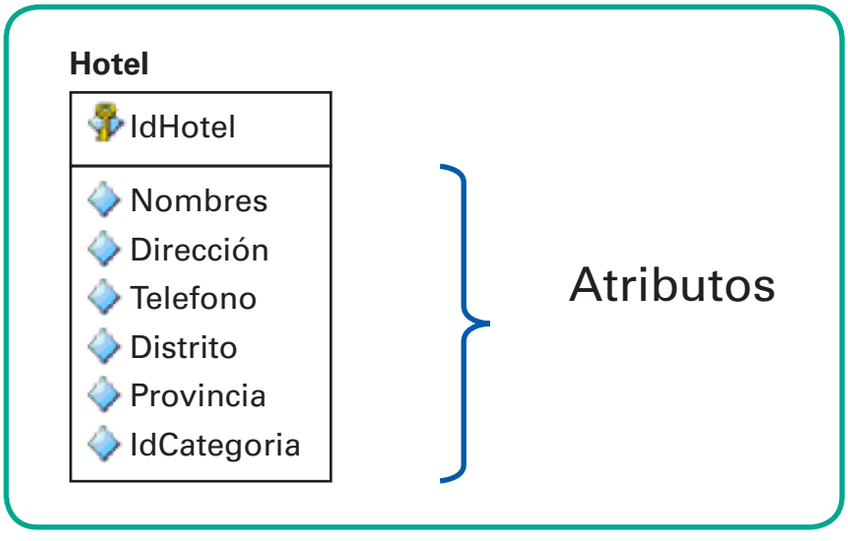


Figura 15: Representación de una tabla del modelo relacional. Fuente: Gómez, 2015.

En la figura 15, se toma como modelo de negocio un sistema administrativo de una cadena de hoteles, y la primera referencia inicial es la tabla Hotel. Para ello, el usuario requiere que se graben en una base de datos la información relacionada con los datos de sus sedes. Entonces, se procede a identificar la clave primaria de la tabla denominándola idHotel; a continuación se identifican los atributos adicionales, tales como nombres, dirección, teléfono, distrito, provincia e IdCategoría. En la figura 16 se puede identificar la clave primaria y la clave foránea de la tabla IdHotel. Cabe mencionar que la clave foránea se genera como producto de la relación entre dos tablas:

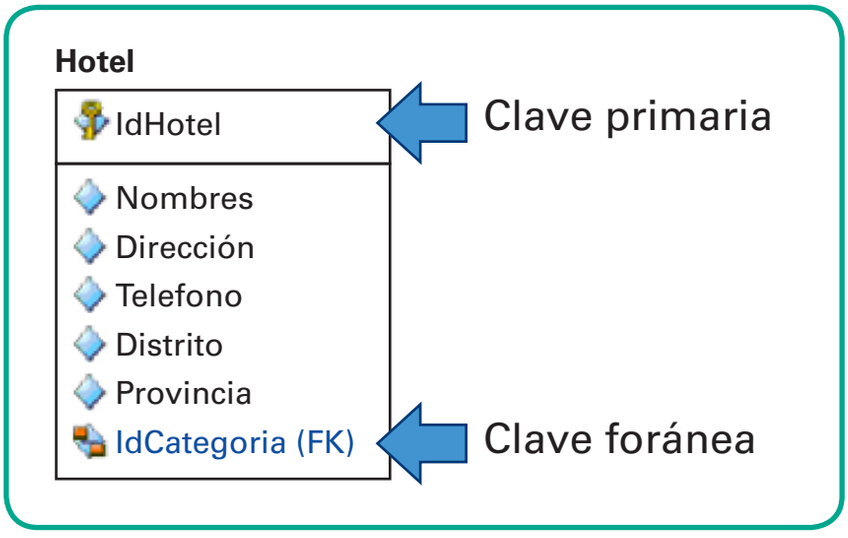


Figura 16: Elementos de una tabla. Fuente: Gómez, 2015.

En la figura 17 se muestra la relación entre dos tablas; para efectos del ejemplo se relacionan la tabla Hotel y la tabla Habitación. La relación entre ellas es de una a muchos; su lectura es la siguiente: en un hotel puede haber muchas habitaciones, muchas habitaciones corresponden a un hotel.

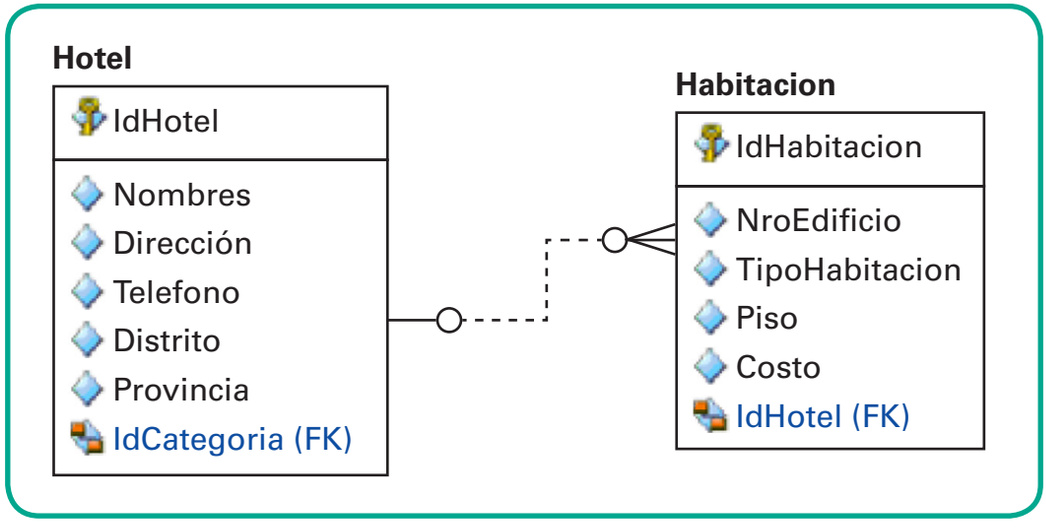


Figura 17: Relación entre dos tablas. Fuente: Gómez, 2015.

2.3. Diagrama de transición de estados

También conocido como DTE, muestra los estados que puede tomar el componente de una aplicación, así como los eventos que implican el cambio de un estado a otro. Los elementos principales de este tipo de diagrama son el estado y la transición.

- Estado: hace referencia al comportamiento del sistema en un determinado período de tiempo y los atributos que lo caracterizan.
- Transición: es el cambio de estado generado por algún evento, muestra el camino del estado inicial al estado final.

De un estado pueden generarse diversas transiciones sobre la base de un evento que desencadena el cambio. Entonces, el diagrama iniciará con un estado inicial como parte del flujo; posteriormente, producto de los eventos se generarán transiciones de otros estados. El estado final será aquel que deje de tener alguna actividad. Cabe mencionar que el diagrama puede tener varios estados finales como parte del flujo, los mismos que se dan a través de las transiciones.

Además de los estados y las transiciones, los diagramas de transición de estados contienen otros componentes, tales como la acción y la actividad. Una acción es una operación que se genera dentro de un estado y se asocia a un evento, la acción se ejecuta al entrar o salir del estado.

Al igual que los diagramas de flujo, los diagramas de estados pueden descomponerse en más de uno para tener una mejor visibilidad de las transiciones; por lo tanto, se pueden tener diagramas de alto nivel y diagramas de bajo nivel. En el caso de que tengamos diagramas de bajo de nivel, los mismos deben ser referenciados en el diagrama principal.

Los elementos del diagrama de estados son los siguientes:

Tabla 3
Notación del diagrama de estados

Representación gráfica	Descripción
	Estado
	Estado inicial
	Estado final
	Transición

Nota: Tomado de Wong, 2017.

A continuación, se mostrará un ejemplo de un diagrama de estados de la revisión de documentos:

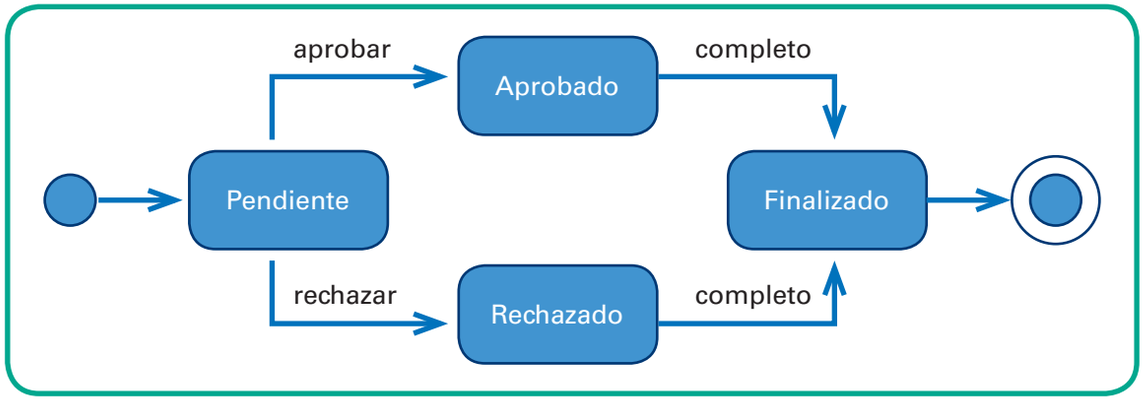


Figura 18: Ejemplo de diagrama de transición de estados. Fuente: Wong, 2017.

2.4. Diccionario de datos

Es una lista de todos los datos que pertenecen a la aplicación. Incluye definiciones de los datos, de modo que todos tengan la misma comprensión de la información; registra la documentación de procesos, almacenes de datos, flujos de datos y datos elementales.

El diccionario de datos debe ir actualizándose continuamente por el equipo de desarrollo, conforme se van dando los mantenimientos al sistema. Esta información es sumamente importante en las etapas de análisis, diseño y construcción del sistema.

El diccionario de datos contiene lo siguiente:

- Estructura de dato: hace referencia al nombre de la entidad.
- Elemento de dato: refiere a los atributos de la entidad.
- Flujos y almacenes de datos: son estructuras de datos.

La notación de los elementos del dato se describirá según lo siguiente:

- Nombre de los datos: se asignará una descripción a la entidad en relación con el significado que este tiene en el contexto del sistema por desarrollar.
- Descripción de los datos: se explica brevemente el significado del dato y su uso en el sistema; la descripción debe ser funcional, de modo que el usuario pueda comprenderla.
- Alias: refiere a la utilización del dato en diferentes contextos dentro del mismo sistema. A fin de identificarlo mejor, el programador podrá asignarle varios nombres de acuerdo con su utilización.
- Longitud: es la cantidad de caracteres que se deben considerar para el registro del dato.
- Rango de valores del elemento del dato: refiere a la cantidad infinita de valores que puede contener un dato. Puede estar en los siguientes rangos:
  - Rango continuo: dentro de un rango válido, el elemento del dato puede tomar cualquier valor.
  - Rango discreto: refiere a los valores limitados que puede tomar el elemento del dato; se restringen los valores.
- Codificación o tipo: se especifica la característica de valor del elemento del dato, de modo que puede ser numérico, alfanumérico, alfabético.

Tabla 4
Ejemplo de Diccionario de datos

Artículo	Detalle del artículo publicado que puede pedirse por las personas que usen LIBSYS	Entidad	30.12.2002
Autores	Los nombres de los autores del artículo que pueden compartir los honorarios	Atributo	30.12.2002
Comprador	La persona u organización que emite una orden de copia del artículo	Entidad	30.12.2002
Honorarios a pagar a	Una relación 1:1 entre Artículo y la Agencia de derechos de autor a quien se debería abonar el honorario de derechos de autor	Relación	29.12.202
Dirección (Comprador)	La dirección del comprador. Ésta se utiliza para cualquier información que se requiera sobre la factura en papel	Atributo	31.12.2002

Nota: Tomado de Sommerville, 2005.

3. Modelado para el desarrollo orientado a objetos

En este modelo, la unidad básica de construcción es la clase y el objeto; el modelo muestra los datos del sistema así como su procesamiento. A continuación, se describen los elementos principales del modelado orientado a objetos:

- Objetos: son instancias de la clase que poseen atributos y servicios. La comunicación entre los objetos se da mediante los mensajes.

- Clases: refiere a los objetos que responden a un mismo mensaje; la clase implementa (encapsula) el método del comportamiento de las instancias.

El modelado orientado a objetos implica la ejecución de las siguientes actividades:

- Identificación de clases, objetos y sus atributos, servicios y operaciones que serán considerados como parte del modelo de desarrollo.

- Descripción del comportamiento de objetos, considerando su estado, transición, evento y acción; asimismo, la colaboración entre ellos.

- Organización de clases por jerarquía a partir de la herencia, a fin de optimizar las propiedades comunes.

- Jerarquización de clases por niveles de abstracción.

Existen varias metodologías orientadas a objeto; entre las más utilizadas se tienen a las siguientes:

Tabla 5
Diferentes métodos de desarrollo de software orientado a objetos

SIGLAS	MÉTODO
RDD	Responsibility-Driven Design
CRC	Tarjetas Clase-Responsabilidad-Colaboración
OOAD	Object-Oriented Análisis and Design
OOD	Object-Oriented Design
OMT	Object Modeling Tecbnique
OOSE	Object Oriented Software Engineerine
OOK/MOSES	Object-Oriented Knowledge
OOSA	Object-Oriented System Analysis
OBA	Object Behavior Analysis
DORA	Object-Oriented Requirements Analysis
Synthesis	Synthesis Method
OOSD	Object Oriented System Development
OOAD/ROSE	Object-Oriented Analysis & Design
FUSION	Object-Oriented Development
UP	Unified Software development Process

Nota: Tomado de Weitzenfeld, 2004.

El proceso unificado de desarrollo de software (Unified Software Development Process) de IBM, hoy en día denominado RUP (Rational Unified Process), junto con el Lenguaje Unificado de Modelado (UML), son la metodología estándar más utilizada para el desarrollo de análisis, diseño e implementación de software. El RUP es un proceso que presenta un conjunto de actividades con una secuencia predefinida, mientras que UML es un lenguaje de modelado.

3.1. Rational Unified Process (RUP)

Aplica las mejores prácticas de los diferentes modelos de procesos de software, y se adapta a proyectos de diversas envergaduras. IBM comercializa el producto, el mismo que ofrece una amplia base de conocimientos como guías y plantillas basadas en el UML; esto permite estandarizar los diagramas del desarrollo de software y, dependiendo del proyecto, aplicar los necesarios. IBM define el RUP así:

> IBM Rational Unified Process®, o RUP®, es una plataforma de proceso de desarrollo de software configurable que entrega mejores prácticas comprobadas y una arquitectura configurable.
>
> Le permite seleccionar y desplegar solamente los componentes de proceso que usted necesita para cada etapa de su proyecto.
>
> La plataforma RUP® incluye (IBM, s/f):
>
> - Herramientas para configurar RUP para las necesidades específicas de su proyecto.
> - Herramientas para transformar sus propios conocimientos internos en componentes del proceso.
> - Herramientas potentes y personalizables de despliegue basadas en web.
> - Una comunidad *online* para intercambio de mejores prácticas con pares y líderes del mercado.

Las fases del RUP se describen en dos dimensiones según lo siguiente:

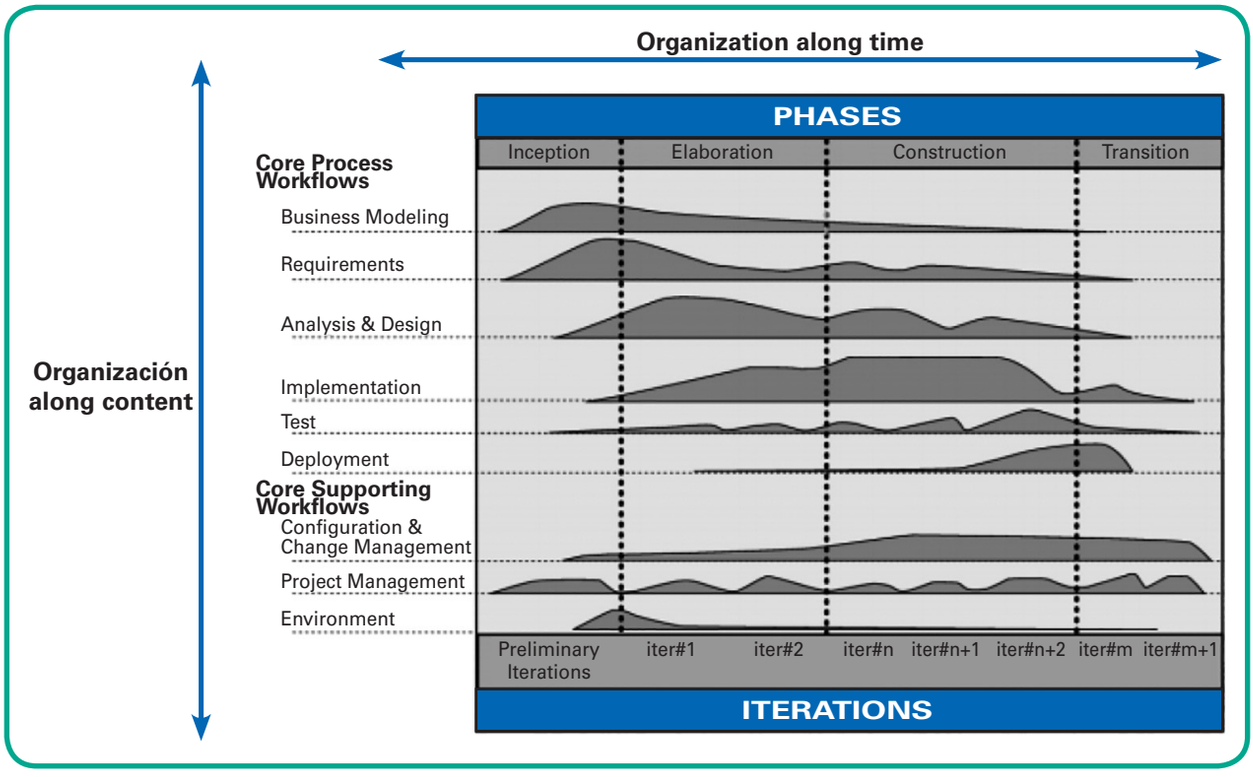


Figura 19: Gráfico del modelo iterativo. Fuente: Rational Software, 1998.

La figura 19 muestra dos perspectivas del modelo:

- Perspectiva dinámica: se muestra en el eje horizontal, representa el tiempo y se expresa en términos de ciclos, fases, iteraciones e hitos.

- Perspectiva estática: se muestra en el eje vertical, se expresa en términos de actividades, artefactos, trabajadores y flujos de trabajo.

Asimismo, se puede observar que cada una de las fases concluye en un hito, por lo que la planificación de proyectos está presente desde las etapas tempranas del desarrollo del software.
RUP muestra el ciclo de vida del producto en cuatro fases:

- Inicio: en esta fase se establece el caso de negocio, se identifican entidades externas (actores) con las que interactuará el sistema; asimismo, el tipo de interacción en cada caso.

- Elaboración: en esta fase se analizan los requisitos funcionales y no funcionales estableciéndose el alcance del sistema, y se define la arquitectura del sistema; asimismo, el plan de proyecto y lista de riesgos del proyecto. En esta fase, de ser necesario, se elaboran pruebas de concepto que muestren la viabilidad del proyecto.

- Construcción: en esta fase se complementa el diseño, se desarrolla e integra el software, se elaboran los manuales de usuario y el documento de pase a producción; asimismo, se trabajan las pruebas internas. La finalidad de esta fase es tener un producto operativo para el usuario final.

- Transición: en esta fase se despliega el software en una ambiente similar al de producción y se efectúan las pruebas necesarias para obtener una versión final del software. Aquí, se pueden identificar incidencias que deben ser corregidas y generar nuevas versiones del producto.

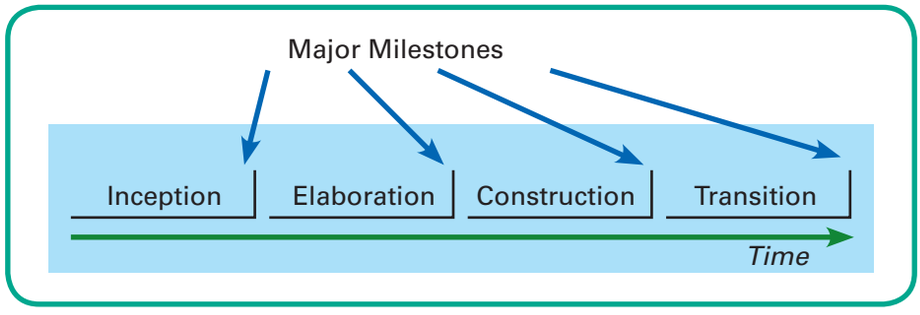


Figura 20: Fases e hitos principales del proceso. Fuente: Rational Software, 1998.

3.2. Lenguaje Unificado de Modelado (UML)

EL UML está respaldado por el OMG (Object Management Group), encargado del mantenimiento de estándares y nuevas versiones del lenguaje unificado. UML proporciona diagramas que esquematizan los requisitos funcionales en términos técnicos que facilitan el desarrollo del software. Dichos diagramas se clasifican jerárquicamente según lo siguiente:

- Diagramas de estructura: hacen referencia a los elementos del sistema.
    - Diagrama de clases
    - Diagrama de componentes
    - Diagrama de objetos
    - Diagrama de estructura compuesta (UML)
    - Diagrama de despliegue
    - Diagrama de paquetes
- Diagramas de comportamiento e interacción: hacen referencia a las interacciones del sistema.
    - Diagrama de actividades
    - Diagrama de casos de uso
    - Diagrama de estados
    - Diagrama de secuencia
    - Diagrama de colaboración
    - Diagrama de tiempos (UML)
    - Diagrama de vista de interacción (UML)

A continuación, se detallan los diagramas más usados en el desarrollo del software.

3.2.1. Diagrama de clases

En este diagrama se muestran las clases de un sistema y la interrelación entre ellas. Se les denomina estáticos porque las clases se denotan junto con sus métodos y atributos.

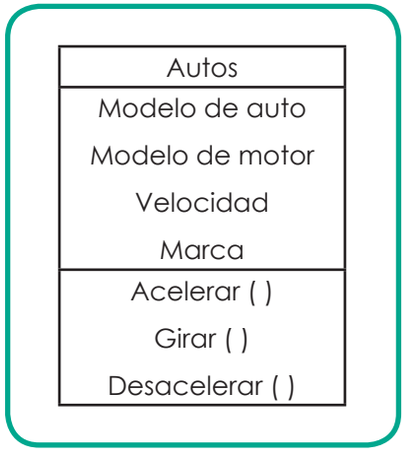


Figura 21: Ejemplo de modelo de clase. Fuente: Wong, 2017.

Las clases se denotan en tres partes: la primera muestra el nombre de la clase; la segunda, los atributos; y la tercera, las acciones.

En el ejemplo tenemos la clase Autos, a continuación sus atributos y, finalmente, las acciones, las cuales, al final de cada una de ellas, muestran un paréntesis, ya que representan funciones que devuelven valores.

3.2.2. Diagrama de casos de uso

Este diagrama describe las acciones que se dan entre el sistema y las entidades externas (actores o sistemas); se describen las actividades funcionalmente para un mejor entendimiento del usuario final. Muestra los escenarios principales y alternativos del sistema.
La notación de los casos de uso es la siguiente:

Tabla 6
Notación de los casos de uso

Representación gráfica	Descripción
	Caso de uso: representa la función del sistema.
	Actor: representa a la entidad externa del sistema.

Nota: Tomado de Wong, 2017.

A continuación, se muestra un ejemplo de diagrama de caso de uso de un sistema de Biblioteca en donde hay tres actores y cuatro casos de uso.

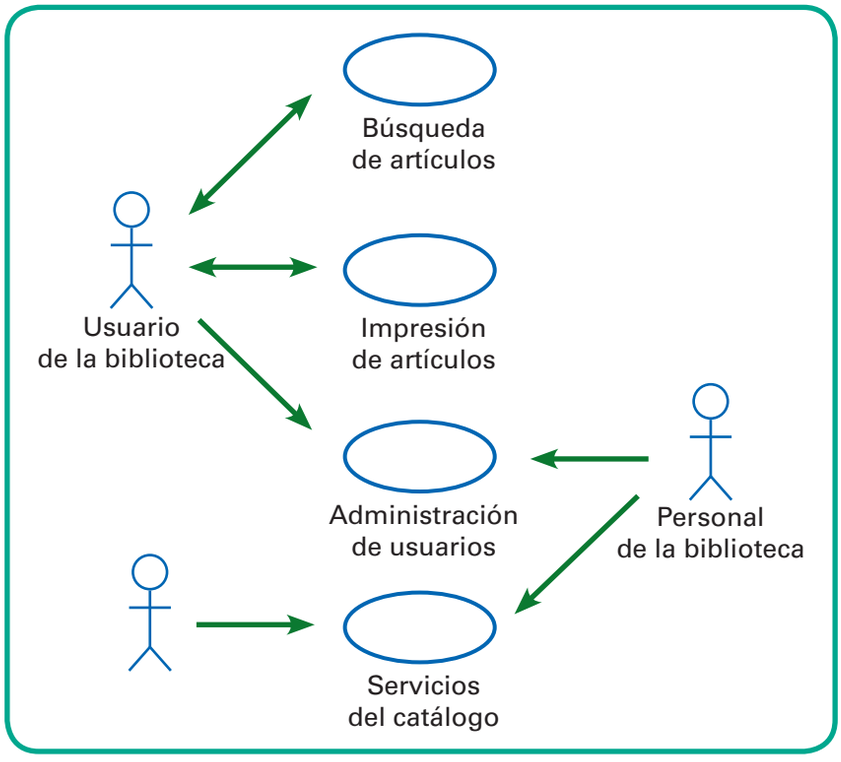


Figura 22: Ejemplo de caso de uso. Fuente: Sommerville, 2005.

3.2.3. Diagramas de secuencia

Muestra la interacción de objetos en el sistema y en el tiempo, y los mensajes que se transmiten entre objetos. A diferencia del caso de uso, que muestra el escenario del flujo principal y escenarios alternativos del sistema, el diagrama de secuencia muestra el detalle del escenario a nivel de actividades.

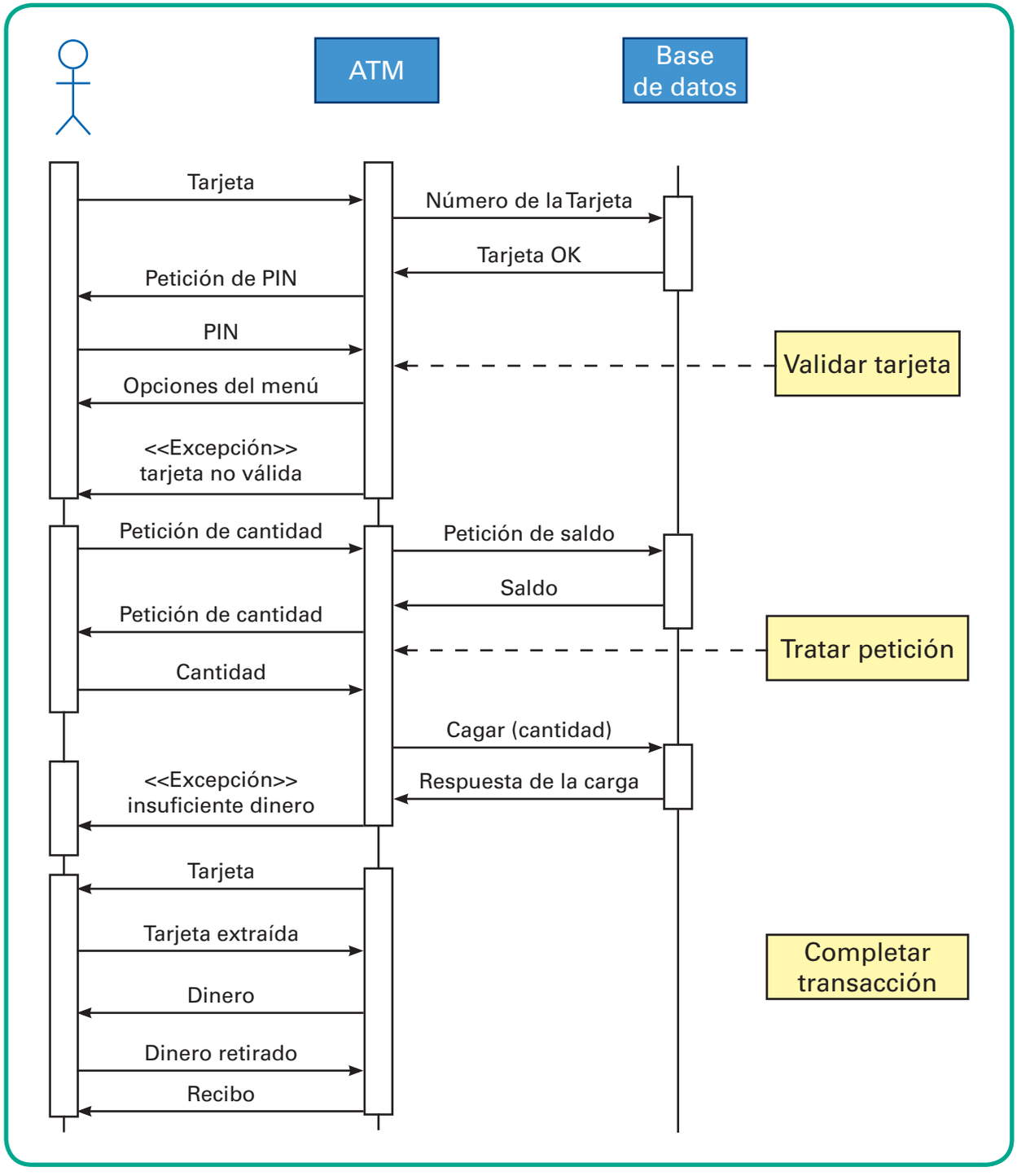


Figura 23: Ejemplo de diagrama de secuencia de retiro de dinero de cajero automático. Fuente: Sommerville, 2005.

El tiempo es representado en forma vertical, cada interacción se incrementa hacia abajo, y los mensajes se remiten entre los objetos mediante flechas.

Lectura seleccionada n.º 1

SWEBOK V3.0. (SWEBOK Guide).
Leer capítulo 5 Software Maintenance apartado 1, 2 y 3, pp. 5-1 a 5-8. 165.

Bourque, P., & Fairley, R. (2014). *SWEBOK V3.0. Guide to the Software Engineering Body of Knowledge*. New Jersey: IEEE. Disponible en http://bit.ly/2zcSdFX

Actividad n.º 1

Foro de discusión sobre el modelamiento de datos.

Instrucciones:

- Ingrese al foro y participe con comentarios críticos y analíticos del tema Modelos de sistemas.
- Lea y analice el tema n.° 1 y 2 del manual
- Responda en el foro a las preguntas acerca del modelado de sistemas:
  - ✓ ¿Cuál es propósito del modelamiento de sistemas?
  - ✓ ¿Cuál es la relación que existe entre el modelamiento de sistemas y el modelamiento de datos?

El modelo de análisis debe cumplir lo siguiente:

- Describir las necesidades del cliente.
- Establecer la base para el desarrollo de un análisis funcional y diseño de software.
- Definir las reglas funcionales y no funcionales que podrán ser validadas por el usuario al culminar el desarrollo de software.

El análisis describe a nivel de sistema lo siguiente:

- Describe el modelo de negocio, transformándolo en requisitos de sistemas que detallan la funcionalidad necesaria para el desarrollo de software y base de datos.
- Se desarrollan los diferentes diagramas especificados en el modelo estructurado o en el modelo orientado a objetos; asimismo, los diagramas de datos que servirán de base para el desarrollo del sistema.
- Se elaboran las interfaces del sistema con base en los estándares especificados de la organización a nivel de diseño. Se recomienda que las mismas sean validadas por el usuario final antes de su desarrollo.
- Detalla la arquitectura a nivel de aplicación de software y datos. Asimismo, el desarrollo y/o consumo de servicios.

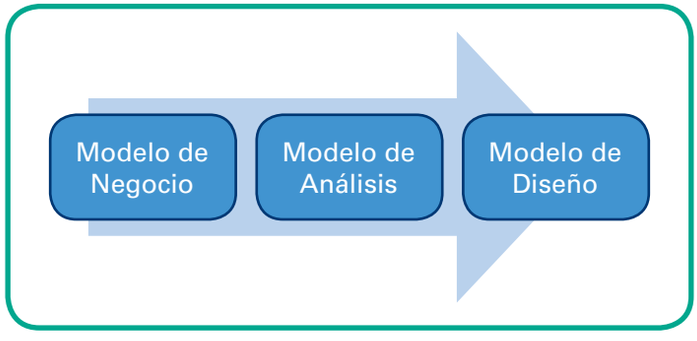


Figura 24: Análisis de software. Fuente: Wong, 2017.

Debe considerarse que la participación activa del usuario en la etapa de análisis es sumamente importante, ya que constituye una garantía de la captura adecuada de que los requerimientos funcionales y no funcionales han sido identificados y descritos de manera correcta. La aprobación del documento de análisis por el usuario garantiza un adecuado desarrollo futuro de la aplicación.

1. Modelado basado en escenarios

Para el modelamiento del análisis de escenarios generalmente se utiliza UML (Unified Modeling Language), que es la técnica empleada por defecto en el modelamiento de entregables. Aquí se trabaja con los diagramas de casos de uso y diagramas de actividades.

2. Modelado orientado al flujo

En este tipo de modelado se utiliza el diagrama de flujo de datos, que detalla el flujo del sistema a nivel de procesos. Cabe mencionar que este tipo de diagrama toma como base el diagrama de contexto y desarrolla a mayor detalle las funcionalidades descritas de cada nivel. El diagrama de flujo de datos también puede complementar a los diagramas UML.

3. Modelado basado en clases

Esta actividad inicia con el desarrollo del caso de uso; a partir de ello se define la lista de objetos que serán denominados como clases del sistema.

Inicialmente, probablemente no se puedan determinar todas las clases; sin embargo, durante el desarrollo del análisis y diseño se podrá ir afinando esta identificación de manera más precisa. Se definen las siguientes clases:

- Clases de entidad: se refieren a la los datos del sistema y que se identifican en los casos de uso.
- Clases de interface de usuario: muestra la interacción con el usuario final.
- Clases de control: muestra las transacciones y control de los objetos definidos en los caso de uso.

Lectura seleccionada n.° 2

Wiegers, K., & Beatty, J. (2013). *Software Requirements* (3a ed.). Washington: Microsoft Press. Disponible en http://bit.ly/2ghNk6M

Actividad n.° 2

Foro de discusión sobre fundamentos de análisis.

Instrucciones:

- Ingrese al foro y participe con comentarios críticos y analíticos del tema modelos de sistemas.
- Lea y analice el tema n.° 3 del manual.
- Responda en el foro a las preguntas acerca del modelado de sistemas:

  1. ¿Cuál es el propósito del modelamiento de análisis?
  2. ¿Cuáles consideras que son las técnicas de modelamiento más utilizadas en la fase de análisis? ¿Por qué?

Tarea académica n.° 1

Genere un caso de uso de un sistema de ventas, y sus flujos alternativos considerando lo siguiente:

- El supermercado Ventas Rápidas requiere vender sus productos y llevar un control de sus ventas; los productos pueden venderse directamente en las cajas de los supermercados en efectivo o mediante tarjeta de crédito Visa o Mastercard. Asimismo, el cliente puede solicitar un *delivery* de productos mediante la página web del supermercado. En este último caso, el cliente debe efectuar el pago directamente vía web con tarjeta VISA o Mastercard antes de que se programe el envío del producto. Para que los clientes puedan efectuar compras por *delivery* deben estar registrados previamente en la base de datos del supermercado.

Enunciado

Complete el documento de **"Plantilla de casos de uso"**.

1. Nombre del caso de uso del sistema		
2. Descripción del caso de uso		
3. Actor(es)		
[Actores que interactúan con el caso de uso.]		
4. Precondiciones		
[Condiciones previas a la realización del caso de uso.]		
5. Poscondiciones		
[Consecuencias luego de la realización.]		
6a. Flujo principal [Pasos que describen la realización del caso de uso. Empieza con la primera acción del actor y el sistema emitirá una respuesta]		
N.°	Acción del actor	Respuesta del sistema
1		
2		
3		
..		
6b. Flujo alternativo [Pasos que describen la realización del caso de uso alternativo]		
N.°	Acción del actor	Respuesta del sistema
1		
2		
3		
..		
7. Requisito asociado (funcional, no funcional)		
8. Prototipo de interfaz de usuario		

I. Rúbrica

nº	Rúbrica	Descripción	Min. puntaje	Máx. puntaje
1	Documento	El trabajo escrito debe presentar todas las pautas expuestas.	0	15
2	Video	Archivo de texto donde se encuentra la URL del video en Youtube, donde se explican los casos de pruebas.	0	5

II. Indicaciones

- Todos los puntos de la rúbrica deberán ser subidos a la plataforma en formato ZIP y como nombre del archivo deberán tener el siguiente formato: **APELLIDO_NOMBRE.zip**
- En caso de plagio, el trabajo será invalidado y la nota será de 00.
- Cualquier duda, consultar al docente por el foro de consultas

Glosario de la Unidad I

A

Atributo: Algunas caracteríticas de una entidad. Puede haber muchos atributos en una entidad (Kendall & Kendall, 2011).

C

Caso de uso: Especificación de un tipo de interacción con el sistema (Sommerville, 2011).

Clase: Una plantilla común para un grupo individual de objetos con atributos comunes y comportamiento común en el análisis y diseño orientado a objetos (Kendall & Kendall, 2011).

D

Diagrama de secuencia: Diagrama que muestra la secuencia de interacciones necesarias para completar alguna operación. En UML, los diagramas de secuencias se pueden asociar con los casos de uso (Sommerville, 2011).

E

Escenario: Modelo de análisis que describe una serie de acciones o tareas que responden a un evento. Cada escenario es una instancia de un caso de uso (International Institute of Business Analysis, 2009).

M

Modelo del ciclo de vida: Marco de referencia que contiene los procesos, actividades y tareas involucradas en el desarrollo, operación y mantenimiento de un producto de software y que abarca toda la vida del sistema desde la definición de sus requerimientos hasta el final de su uso (Indecopi, 2006).

O

Objeto: En el enfoque orientado a objetos, un objeto es una representación por computadora de algo o cosa del mundo real; puede tener atributos y comportamientos (Kendall & Kendall, 2011).

R

Requerimiento: Una condición o capacidad requerida por una parte interesada para resolver un problema o alcanzar un objetivo. Una condición o capacidad que debe cumplir un componente de solución o solución para satisfacer un contrato, norma, especificación, u otros documentos formalmente impuestos (International Institute of Business Analysis, 2009).

U

UML (Unifed Modeling Language): Proporciona un conjunto estandarizado de herramientas para documentar el modelo orientado a objetos en el análisis y diseño de un sistema de software (Kendall & Kendall, 2011).

Bibliografía de la Unidad I

Arisholm, E., Gallis, H., Dyba, T., & Dag I.K. Sjoberg. (2007). Evaluating pair programming with respect to system complexity and programmer expertise. *IEEE Transactions on Software Engineering, 33*(2), 65-86. Disponible en http://ieeexplore.ieee.org/document/4052584/

Bourque, P., & Fairley, R. (2014). *SWEBOK V3.0. Guide to the Software Engineering Body of Knowledge*. New Jersey: IEEE. Disponible en http://bit.ly/2zcSdFX

IBM (1998), Rational Unified Process Best Practices for Software Development Teams. Recuperado de https://www.ibm.com/developerworks/rational/library/253.html

IBM. (s/f). *Plataforma para desarrollo de software*. Lima: autor. Disponible en https://ibm.co/2y5q05B https://www-01.ibm.com/software/pe/rational/rup.html

Kendall, K. & Kendall, J. (2013). *Systems analysis and design* (9a ed.). New Yersey: Prentice Hall.

Pressman, R. (2005). *Ingeniería del Software. Un enfoque práctico* (6a ed.). D.F., México:McGraw-Hill.

Sommerville, Ian. (2011). *Software Engineering* (9a ed.). Madrid: Pearson Education.

Wiegers, K., & Beatty, J. (2013). *Software Requirements* (3a ed.). Washington: Microsoft Press. Disponible en http://bit.ly/2ghNk6M

Autoevaluación n.º 1

1. Marque la respuesta correcta: ¿Cómo se le llama a la persona que es dueña del modelamiento de sistemas, análisis y desarrollo de software?
   a) Cliente
   b) Propietario
   c) Usuario
   d) Proveedor
   e) Analista

2. Seleccione la alternativa correcta en relación con los tipos de modelos de sistemas.
   a) Modelo de contexto
   b) Modelo de base de datos
   c) Modelo de diseño
   d) Modelo de algoritmos
   e) Modelo de análisis

3. Seleccione la alternativa correcta en relación con los tipos de modelo de datos:
   a) Flujo de datos
   b) Relacional
   c) Base de datos
   d) Entidad–relación
   e) Atributos

4. ¿La creación de escenarios en qué fase del ciclo de vida se utiliza?
   Seleccione la alternativa correcta.
   a) Diseño
   b) Programación
   c) Análisis
   d) Calidad
   e) Pase a producción

5. Seleccione la afirmación correcta en relación con el método estructurado:
   a) Proporciona un marco para el modelado detallado de sistemas como parte de la elicitación y análisis de requerimientos.
   b) Muestra una perspectiva funcional en donde cada transformación representa un único proceso o función.
   c) Se utilizan para describir el comportamiento del sistema en su totalidad.
   d) Describe cómo responde un sistema a eventos internos o externos.
   e) Muestra las entidades de datos, sus atributos asociados y las relaciones entre estas entidades.

Seleccione la alternativa correcta.

6. ¿Cuál es la diferencia entre el modelo relacional y el modelo entidad-relación?.
   a) La diagramación del modelo relacional es mediante entidades y atributos, la diagramación del modelo entidad-relación es mediante tablas.
   b) La diagramación del modelo relacional es mediante entidades y atributos, la diagramación del modelo entidad-relación es mediante datos.
   c) La diagramación del modelo relacional es mediante tablas, la diagramación del modelo entidad-relación es mediante entidades y atributos.
   d) La diagramación del modelo relacional es mediante cardinalidades, la diagramación del modelo entidad-relación es mediante datos.
   e) La diagramación del modelo relacional es mediante entidades, la diagramación del modelo entidad-relación es mediante una base de datos.

7. Un escenario es parte de:
   a) Caso de uso
   b) Entidad
   c) Atributo
   d) Especificación
   e) Requerimiento

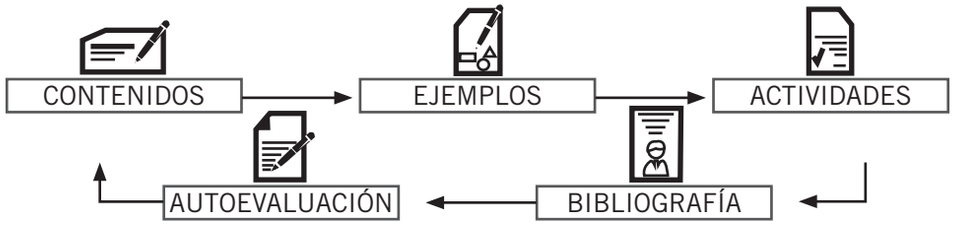
8. Muestra la relación entre los actores y el caso de uso:
   a) Diagrama de flujo
   b) Diagrama de secuencia
   c) Diagrama de clases
   d) Diagrama de caso de uso
   e) Diagrama de estado

9. Muestra los objetos que participan en la interacción y la secuencia de mensajes que intercambian:
   a) Diagrama de flujo
   b) Diagrama de secuencia
   c) Diagrama de clases
   d) Diagrama de caso de uso
   e) Diagrama de estado

10. Las cardinalidades de un modelo relacional o entidad-relación pueden ser:
    a) 2 tipos
    b) 3 tipos
    c) 4 tipos
    d) 5 tipos
    e) 1 tipo

UNIDAD II
ELICITACIÓN DE REQUERIMIENTOS

DIAGRAMA DE PRESENTACIÓN DE LA UNIDAD II



ORGANIZACIÓN DE LOS APRENDIZAJES

RESULTADO DE APRENDIZAJE: Al finalizar la unidad, el estudiante será capaz de elicitar los requerimientos de soft-ware para su proyecto.

CONOCIMIENTOS	HABILIDADES	ACTITUDES
Tema n.° 1: Fundamento de los requerimientos 1. Análisis de negocio 2. Ingeniería de requisitos 2.1. Inicio 2.2. Obtención 2.2.1. Problemas de ámbito 2.2.2. Problemas de comprensión 2.2.3. Problemas de volatilidad 2.3. Elaboración 2.4. Negociación 2.5. Especificación 2.6. Validación 2.7. Gestión de requisitos **Tema n.° 2: Origen de los requerimientos** 1. Definición de requisito 2. Esquema de clasificación de requisitos 2.1. Requisitos de negocio 2.2. Requisitos de partes interesadas 2.3. Requisitos de la solución 2.3.1. Requisitos funcionales 2.3.2. Requisitos no funcionales 3. Requisitos de transición **Lectura seleccionada n.° 3** *BABOK Guide v2.0* (pp. 99-117). **Tema n.° 3: Técnicas de elicitación de requerimientos** 1. Tormenta de ideas (*Brainstorming*) 2. Análisis de documentos 3. *Focus group* 4. Análisis de interfaces 5. Entrevistas 6. Observación 7. Prototipos 8. Talleres de requisitos 9. Encuestas/cuestionarios 10. Técnicas grupales de toma de decisiones 11. Estudios comparativos **Lectura seleccionada n.° 4** *Definición de perfiles en herramientas de gestión de requisitos* (Mcdonald, 2005). **Autoevaluación de la Unidad II**	1. Administrar la obtención de los requerimientos de software. **Actividad n.° 1** Los estudiantes participan en el foro de discusión sobre modelos de sistemas y modelos de análisis. **Tarea académica n.° 2**	1. Concuerda con su equipo de trabajo la priorización de los requerimientos de software.

Fundamento de los requerimientos

Tema n.° 1

En el siguiente capítulo, usted comprenderá los diferentes conceptos de análisis de negocio, ingeniería de requisitos y técnicas de elicitación, que servirán de base para la obtención de requerimientos de la fase inicial del ciclo de vida.

Los requerimientos de software surgen de la necesidad de automatización de un proceso manual; sin embargo, muchas veces a pesar de que el usuario sabe y conoce su proceso, trasladar su idea a un producto de software toma mucho tiempo; por ello, como parte del ciclo de vida del desarrollo del software se han generado algunas ramas de especialización, tales como el análisis de negocio y la ingeniería de requisitos, las cuales se orientan a obtener una definición concreta de la necesidad del usuario para luego procesarla y generar un producto de software que cubra las expectativas del cliente. A continuación, mostraremos el detalle de estas definiciones.

1. Análisis de negocio

El International Institute of Business Analysis (2009) define lo siguiente en relación con el análisis de negocio:

> El análisis de negocios es el conjunto de tareas y técnicas utilizadas para trabajar como enlace entre las partes interesadas a fin de comprender la estructura, las políticas y las operaciones de una organización y recomendar soluciones que permitan a la organización alcanzar sus objetivos (p. 3)

El análisis de negocio consigna el desarrollo del modelo de proceso del negocio. En esta fase, el analista de negocios busca un entendimiento del negocio de la organización; no considera aún el requerimiento en sí, solo busca conocer el sector de negocio que, por ejemplo, puede ser banca, retail, telecomunicaciones, gobierno, minería, entre otros; luego buscará comprender con más detalle el área que necesita una mejora y definirá el requerimiento específico por automatizar. El conocimiento del negocio implica conocer a la organización, cómo funciona, cuál es su meta, visión, qué productos ofrece, fortalezas y debilidades. Asimismo, quiénes son los usuarios finales de los productos o servicios y canales de comunicación. Si consideramos que hoy en día las redes sociales y las aplicaciones web y móviles han cobrado mayor relevancia, el canal de comunicación hacia el cliente es un factor crítico que debe ser analizado por el analista de negocios.

En algunas ocasiones, las metas, objetivos o visión de la organización no se encuentran bien definidos, lo que dificulta el trabajo del analista de negocio. Toda acción de mejora identificada, indicadores de desempeño, debe estar alineada con las metas y objetivos de la organización, por lo cual a veces esta es la primera tarea del analista de negocio.

Las oportunidades de mejora identificadas dentro de las unidades de negocio deben estar alineadas a los objetivos y metas de negocio de la organización; en ese sentido, los directivos tomarán las decisiones para la priorización de requerimientos de automatización.

El International Institute of Business Analysis (IIBA) es la asociación internacional que mantiene el estándar de mejores prácticas para la ejecución de análisis de negocios, tiene más de 29 000 miembros a nivel internacional, provee una certificación para todos los profesionales que deseen especializarse en análisis de negocios, y provee una Guía, que es el BABOK, en donde se pueden visualizar todas las mejores prácticas. Asimismo, tiene una comunidad que apoya y colabora activamente para enriquecer esta guía.

Project Management Institute, una de las organizaciones más grandes de profesiones de dirección de proyectos, dentro de sus prácticas incluye un área de proceso de recopilación de requisitos, la misma que difunde las mejores prácticas para la obtención de requisitos desde etapas tempranas. Incluye, entre ellas, herramientas que ayudan a la elicitación de requisitos y recalca la importancia de las mismas en el desarrollo de los proyectos y de los productos de software que se generen como parte de las mejoras propuestas en la organización. Asimismo, hace énfasis en la alineación estratégica que debe existir entre las oportunidades de mejora y los objetivos estratégicos de la organización.

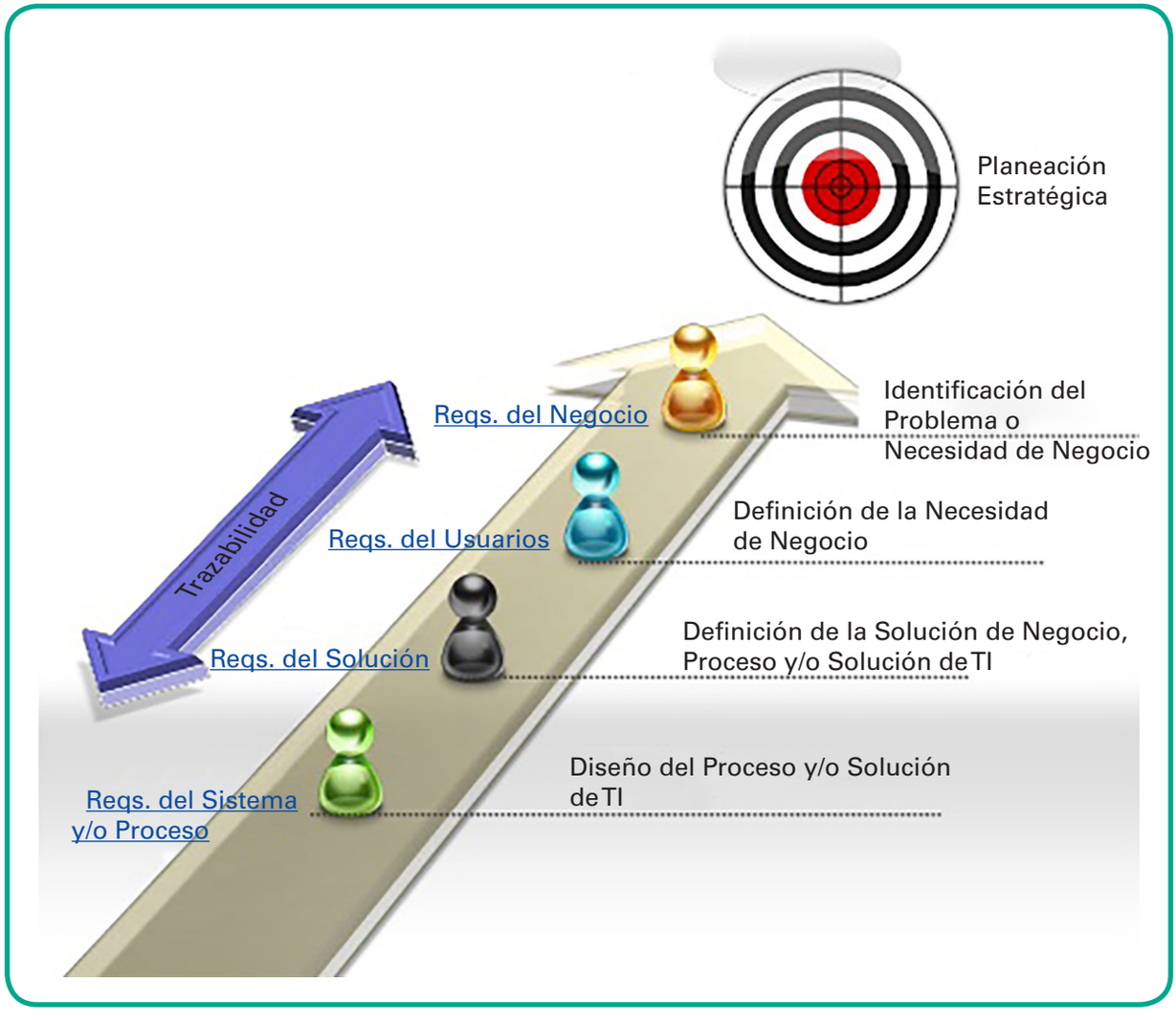


Figura 25: Análisis de negocio. Fuente: Almeida, 2012.

El gráfico describe la perspectiva de la gestión de análisis de negocios para la generación de definiciones de soluciones de TI. También, muestra cómo todos los requisitos, desde las unidades de negocio hasta el software generado, deben obedecer a los objetivos planteados en la Planeación estratégica organizacional.

2. Ingeniería de requisitos

La ingeniería de requisitos ayuda a plasmar los requisitos funcionales de los usuarios en diagramas técnicos que permitan a los analistas y programadores de software un mejor entendimiento del producto de software que tienen que desarrollar. En esta primera etapa, los diagramas de contexto y diagramas de flujo de datos son los primeros por elaborar para poder lograr un entendimiento con el usuario de lo que se requiere.

La ingeniería de requisitos constituye la fase inicial del desarrollo de software, pues ayuda a generar las especificaciones preliminares para tal desarrollo.

En esta primera etapa pueden elaborarse incluso pruebas de concepto preliminares para evaluar la tecnología por aplicar y la factibilidad de continuar con el desarrollo del software. Igualmente, se aplican técnicas de experiencia de usuario para que el usuario en esta primera etapa conozca a alto nivel cómo quedarán sus interfaces. El riesgo en esta etapa es la indecisión del usuario, pues si aún no sabe concretamente lo que quiere o tiene mucha dificultad para tomar decisiones, esta etapa puede tomar mucho tiempo.

Pressman (2005) precisa que

> La ingeniería de requisitos se lleva a cabo a través de siete funciones que en algunos casos pueden ejecutarse de manera paralela, sin embargo las mismas se adaptarán a la metodología a utilizar por la organización y a la necesidad del proyecto, la finalidad es definir claramente la necesidad del usuario final de modo que pueda desarrollarse adecuadamente un análisis, diseño y construcción que refleje lo que el usuario solicitó (p. 157).

Las funciones que define Pressman se muestran en el siguiente gráfico:

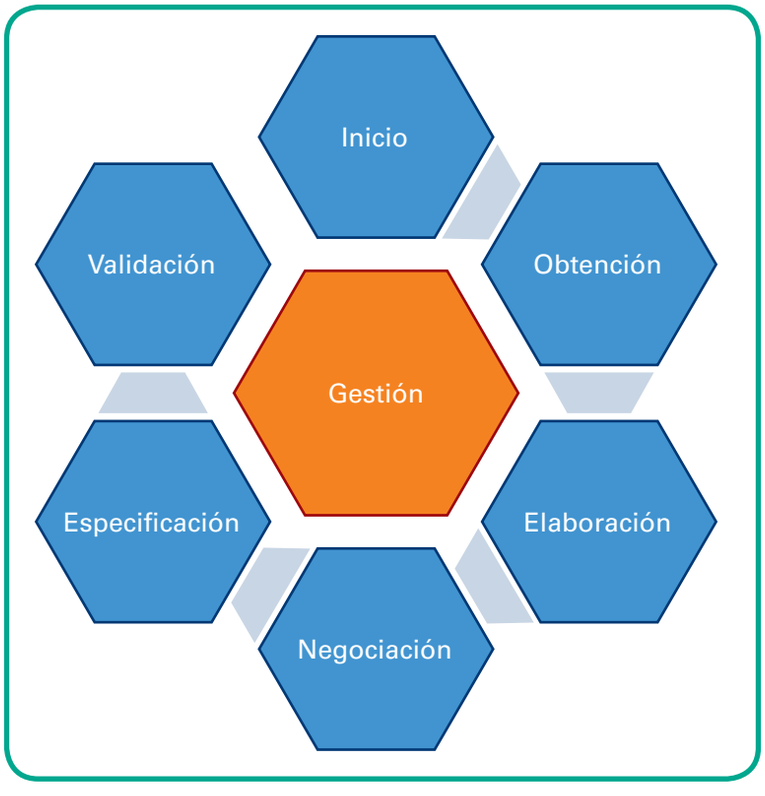


Figura 26: Funciones de la ingeniería de requisitos según Pressman. Fuente: Wong, 2017.

2.1. Inicio

La mayoría de proyectos inician con la identificación de una necesidad de negocio, mercado nuevo o servicio potencial. Luego de haber comprobado la factibilidad del proyecto, el personal funcional y técnico del proyecto efectúa preguntas libres al cliente con el objetivo de obtener una comprensión básica del problema y establecer una solución de alto nivel; esta puede constituir la primera comunicación con el cliente.

2.2. Obtención

Se listan los problemas que dificultan de la obtención de los requisitos; en ese sentido, se clasifican como problemas de ámbito, de comprensión y volatilidad.

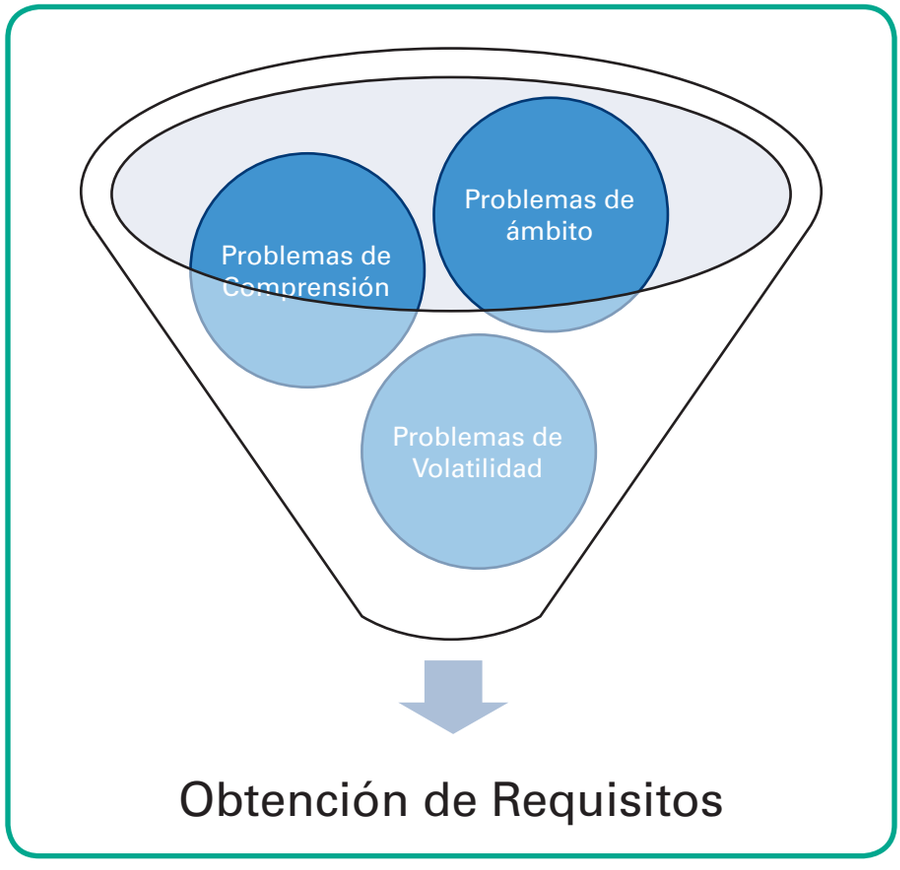


Figura 27: Problemas de la obtención de requisitos. Fuente: Wong, 2017.

A continuación, se detallan cada uno de ellos:

2.2.1. Problemas de ámbito

Se catalogan de esta manera cuando el alcance del sistema no está bien definido; los usuarios hacen referencia a diferentes temas sin establecer claramente los límites del sistema. En este caso, es necesario que el analista establezca los límites mediante supuestos y restricciones, definiendo aquello que será parte del sistema y aquello que no será parte del sistema.

2.2.2. Problemas de comprensión

El usuario final no sabe concretamente lo que necesita, no tiene conocimiento de elementos tecnológicos, es muy funcional. También entran en esta categoría los usuarios que no saben transmitir claramente sus necesidades o la indican a muy alto nivel porque no disponen de mucho tiempo; por tanto, omiten información necesaria para el desarrollo del sistema. Existen usuarios que también especifican requisitos que corresponden a otras unidades de negocio o a otros sistemas sin una coordinación previa a nivel funcional.

2.2.3. Problemas de volatilidad

Existen casos en donde los usuarios por cada sesión de relevamiento de información en lugar de complementar, modifican la información antes precisada para el desarrollo del sistema; por tanto, es necesario que se concreten actas de trabajo en donde se especifiquen los temas tratados y acuerdos. Esta actividad también se puede dificultar cuando los usuarios funcionales cambian de una reunión a otra; por ello, es necesario que se solicite la interacción con un usuario representativo que pueda expresar las necesidades de la unidad, organizaciones y verifique el producto culminado.

2.3. Elaboración

En esta etapa se desarrolla el diseño técnico y las reglas de negocios, se acota el alcance, y se definen las restricciones del software. Asimismo, se elaboran los escenarios y los flujos principales y alternativos del software. Esta etapa puede ser soportada por los diagramas descritos en el UML.

2.4. Negociación

Resulta común que durante la especificación y desarrollo de software se encuentren muchos interesados en el proyecto y cada uno de ellos con una visión y perspectiva diferente del desarrollo del requerimiento, razón por la cual se deberá conciliar con los interesados y solicitar aprobadores específicos de nivel funcional y técnico; con ellos se deberán evaluar los riesgos, estimaciones de proyecto, costos y plazos de entrega.

2.5. Especificación

Algunos autores sugieren que el desarrollo de la especificación del requerimiento debe ser plasmado en una plantilla estándar; sin embargo, la captura de la especificación de requerimientos debe ser flexible y se deben aplicar las técnicas de elicitación necesarias a fin de lograr un entendimiento entre los analistas de sistemas y el usuario final. En todos los casos las buenas prácticas refieren al desarrollo de gráficos funcionales que permitan al usuario visualizar de manera gráfica su proceso y validarlo. Igualmente, el desarrollo de prototipos le permitirá visualizar de manera anticipada las interfaces del sistema solicitado.

2.6. Validación

Refiere a una actividad de calidad que revisa las especificaciones de los requisitos del software. Esta etapa sirve para validar previamente si los requisitos cumplen con las especificaciones del usuario. Si producto de esta revisión se detectan no conformidades, la validación deberá revisar si las mismas han sido levantadas, antes de iniciar el proceso de desarrollo del software. Muchas veces este proceso de validación de los productos de trabajo es efectuado por el líder del proyecto y luego por el usuario final; con ello se establece el compromiso de que el desarrollo contemplará todo aquello que ha sido solicitado por el cliente.

2.7. Gestión de requisitos

Los requisitos de un sistema pueden variar, se puede ampliar, reducir o modificar el alcance inicial, razón por la cual es necesario establecer una bitácora de cambios de los mismos, ya que durante todo el ciclo de vida del desarrollo del software, este puede sufrir alteraciones.

Origen de los requerimientos

Tema n.° 2

Los requerimientos de software nacen a raíz de una necesidad de una automatización, en la mayoría de los casos se describe una regla funcional a alto nivel y se describe una idea de manera global. Con la finalidad de definir a mayor detalle los requerimientos, se describen mediante reglas de negocios y, en muchos casos, se hacen uso de los diagramas de procesos para que el personal de desarrollo pueda entender con un mayor nivel de detalle las especificaciones funcionales. El personal de sistemas es el encargado de trasformar los requerimientos funcionales en requerimientos de sistemas y, de esta manera, generar un nivel más técnico de especificaciones para el desarrollo del software.

1. Definición de requisito

El International Institute of Business Analysis [IIBA] (2009) define el término *requisito* según lo siguiente:

> The term "requirement" is one that generates a lot of discussion within the business analysis community. Many of these debates focus on what should or should not be considered a requirement, and what are the necessary characteristics of a requirement. When reading the BABOK® Guide, however, it is vital that "requirement" be understood in the broadest possible sense. Requirements include, but are not limited to, past, present, and future conditions or capabilities in an enterprise and descriptions of organizational structures, roles, processes, policies, rules, and information systems. A requirement may describe the current or the future state of any aspect of the enterprise. (p. 005).

Muchos autores que escriben en relación con el análisis de negocios, precisan que los requisitos hacen referencia al desarrollo de un sistema de información; otros indican que se deben incluir funciones empresariales futuras. Lo importante es definir específicamente la mejora, la cual puede considerar el perfeccionamiento de un proceso, el desarrollo de sistema, una automatización, entre otros. En el caso de que se considere una mejora tecnológica, es importante tener en cuenta la evolución de la misma y el tiempo de vida de la solución, pues con el constante cambio tecnológico actual, puede ser que cuando la solución salga al mercado ya sea obsoleta; por tanto, los cambios tecnológicos implicarán acciones a corto plazo.

2. Esquema de clasificación de requisitos

El IIBA muestra la siguiente clasificación para describir los requisitos:

- Requisitos de negocio
- Requisitos de las partes interesadas
- Requisitos de la solución
- Requisitos de transición

La clasificación de requisitos considera diferentes enfoques porque es necesario cubrir todas las perspectivas de lo requerido por el usuario y, de esta manera, poder generar una propuesta adecuada (p. 5).

A continuación, se detallan cada uno de ellos:

2.1. Requisitos de negocio

Se describen a alto nivel las necesidades de la organización, sus metas y objetivos. Esto implica precisar los antecedentes y objetivos del proyecto, indicadores de desempeño de gestión y operación para el seguimiento del proyecto.

Algunos de los requisitos de negocio son:

- Debemos ser líderes en el mercado en el uso de aplicaciones móviles para las gestiones de nuestras operaciones.
- El 40% de las operaciones generadas por nuestros clientes debe ser gestionada por nuestra página web.

2.2. Requisitos de las partes interesadas

Son las declaraciones de las necesidades de un interesado o clase particular de partes interesadas. Describen las necesidades que tiene un determinado actor y cómo interactúan las partes interesadas con la solución. El análisis de requisitos describe los requerimientos de dichas partes.

A modo de ejemplo, podemos mencionar dos requisitos de partes interesadas:

- El estatus de contratación de personal debe ser visualizado por cada una de las unidades de negocio.
- El área de tesorería debe tener acceso a la información de pagos que se generan por órdenes de compra en el área de logística.

2.3. Requisitos de la solución

Describe las características de una solución que cumpla con los requerimientos del negocio y los requerimientos de las partes interesadas. Se desarrollan y definen a través del análisis de requisitos. Con frecuencia se dividen requisitos funcionales y no funcionales, especialmente cuando los requisitos describen una solución de desarrollo de software.

2.3.1. Requisitos funcionales

Describen el comportamiento y los datos que el sistema administrará. También, las capacidades que el sistema podrá realizar en términos de comportamientos o acciones o respuestas de aplicación de tecnología de la información específicas de las operaciones.

Algunos ejemplos de requisitos funcionales son:

- El sistema debe permitir el registro de todos los clientes de la tienda.
- El sistema debe permitir el registro de las marcas de los productos y el movimiento logístico del almacén y la tienda.
- El sistema debe permitir la generación de la boleta de venta y factura.
- Generación del cálculo automático de IGV

2.3.2. Requisitos no funcionales

Este tipo de requisitos están relacionados con la operación del software y hacen referencia al comportamiento de la solución, describen condiciones ambientales bajo las cuales la solución debe permanecer activa por determinados períodos de tiempo o cualidades que los sistemas deben tener a nivel operacional. Los requisitos no funcionales están relacionados con la capacidad, experiencia de usuario, disponibilidad, velocidad, seguridad y arquitectura de la información.

A continuación se muestran ejemplos de requisitos funcionales:

- El sistema debe funcionar en Explorer 7 y 8, también en Chrome.
- El sistema web debe tener disponibilidad permanente para los usuarios; debe ser 7 x 24.

- Las interfaces del aplicativo móvil deben ser amigables e intuitivas.
- El nivel de respuesta de los reportes del sistema no debe ser mayor a 30 segundos.

2.4. Requisitos de transición

El IIBA describe lo siguiente:

> Las capacidades que debe tener la solución para facilitar la transición desde el estado actual de la empresa a un estado futuro deseado, pero que no será necesario una vez que la transición esté completa. Se diferencian de otros tipos de requisitos porque son siempre de naturaleza temporal y porque no pueden desarrollarse hasta que se definen una solución nueva y existente. Normalmente cubren la conversión de datos de los sistemas existentes, las brechas de habilidades que deben ser abordadas y otros cambios relacionados para alcanzar el estado futuro deseado. Se desarrollan y definen mediante la evaluación y validación de soluciones (p. 006).

Como ejemplos de requisitos de transición pueden mencionarse los siguientes:

- Antes de ejecutar el pase a producción se deben correr los bacheros (sistemas por lotes) de la migración de datos, ya que sin ellos no se podrá tener información para efectuar las pruebas de la aplicación.
- El pago de los empleados se hace mediante el servicio de VISA con el banco, por tanto nuestro sistema de recursos humanos debe interconectarse con ese servicio antes de difundir al personal al que ya se abonó su pago.
- La implementación de la nueva tecnología implica la capacitación previa de los desarrolladores; asimismo el acompañamiento de la empresa para la elaboración del producto.

Lectura seleccionada n.º 3

A Guide to the Business Analysis Body of Knowledge (BABOK Guide) v2.0.

Leer el capítulo 6, "Requirements Analysis", pp. 99-120. Disponible en http://bit.ly/2hzVszh

Actividad n.º 3

Foro de discusión sobre el modelamiento de datos.

Instrucciones:

- Ingrese al foro y participe con comentarios críticos y analíticos del tema modelos de sistemas.
- Lea y analice los temas 1 y 2 del manual.
- Responda en el foro a las preguntas acerca de los fundamentos de los requerimientos:

  Mencione cuatro requisitos funcionales y dos requisitos no funcionales por considerar en un sistema de biblioteca.

Técnicas de elicitación de requerimientos

Tema n.° 3

La actividad de elicitación de requisitos constituye una actividad clave dentro del ciclo de vida de desarrollo del producto; esto debido a que es la base para la definición del alcance de la mejora por efectuar dentro la organización. El éxito de la mejora residirá en la participación activa de los usuarios y en la definición asertiva de los requisitos.

The International Institute of Business Analysis (2009) en su *Guía de los fundamentos del conocimiento del análisis del negocio* (A Guide to the Business Analysis Body of Knowledge: BABOK) menciona las técnicas de elicitación más aceptadas, que reseñamos a continuación.

1. Lluvia de ideas (*Brainstorming*)

Esta técnica, según el PMI, está catalogada como una técnica grupal de creatividad, es utilizada para generar y recopilar varias ideas relacionadas con un mismo tema. Para efectos de la gestión de requisitos, se utiliza para generar ideas relacionadas con los requisitos del producto para, posteriormente, profundizar a mayor detalle en el tema. Esta técnica es efectuada por un facilitador que dirige al grupo. Las sesiones pueden ser abiertas o estructuradas con la finalidad de refinar las definiciones. Esta técnica no establece orden de prioridades; sin embargo, pueden complementarse con otras técnicas que sí lo hacen.

Figura 28: Proceso de lluvia de ideas. Disponible en http://bit.ly/2wDkeVo

El gráfico muestra un proceso de lluvia de ideas estructurado y guiado por facilitadores. Se aprecia la motivación para la generación de ideas, las reglas de redacción de las ideas y el ordenamiento para llegar a un consenso de una idea general.

La lluvia de ideas para su clasificación puede complementarse con otras técnicas grupales de creatividad, tales como:

- Técnicas de grupo nominal, que mediante la votación y jerarquización seleccionan las ideas más útiles.
- Mapa conceptual o mental, que consolidan la información de la lluvia de ideas, y la clasifican en ideas que tienen puntos en común e ideas que tienen diferencias. La clasificación sirve como base para la generación de nuevas ideas pero de manera más estructurada en su segunda iteración.
- Diagramas de afinidad, que clasifica a las ideas de acuerdo con su similitud.
- Análisis de decisiones con múltiples criterios, que clasifica a la información con base en una serie de criterios.

2. Análisis de documentos

Esta técnica involucra el análisis de la documentación existente que pueda servir de base para el análisis de requisitos, se busca identificar información relevante para obtener requisitos, involucra la revisión de documentación de productos similares, planes de negocios, estudios de mercado, contratos, solicitudes de propuestas, memorandos, directivas existentes, procedimientos, guías de capacitación, entre otros.

El objetivo es obtener la mejor cobertura de los requisitos y, de esta manera, describir correctamente el producto por desarrollar.

3. *Focus group*

Un grupo focal se genera para intercambiar ideas acerca del producto, servicio o resultado propuesto. El grupo es guiado por un facilitador y los participantes son expertos en el tema por tratar, se genera un espacio para discutir del tema y que cada uno exprese su perspectiva basado en su experiencia. También pueden participar otras personas pero limitadas solo a observar y no a participar activamente de la sesión.

El facilitador tiene la misión de guiar al grupo para obtener un resultado cualitativo satisfactorio y generar el informe final.

Esta técnica de elicitación puede ser utilizada en cualquier fase del ciclo de vida del desarrollo del software, ya sea análisis, diseño, construcción, pruebas o antes del pase a producción; en este caso el tema de discusión del grupo estará relacionado con los requisitos establecidos previamente, con el objetivo de aclarar, modificar el requisito existente o generar nuevos requisitos

4. Análisis de interfaces

La interfaz es el punto de interconexión entre dos partes del sistema. Los sistemas poseen más de una interfaz en su desarrollo. Las interfaces se catalogan de la siguiente manera:

- Interfaces de usuario: refiere a los usuarios que interactúan con el sistema, ya sea en el registro de información, operaciones o consulta. A cada interfaz de interacción se le considera una interface de usuario. Considerando que el usuario tendrá un papel relevante en su uso, será necesario que estas sean previamente revisadas y validadas por el usuario antes de su elaboración.

En la figura 29 puede apreciarse, a modo de ejemplo, la interacción directa del usuario con el sistema, a nivel de la operación del sistema.

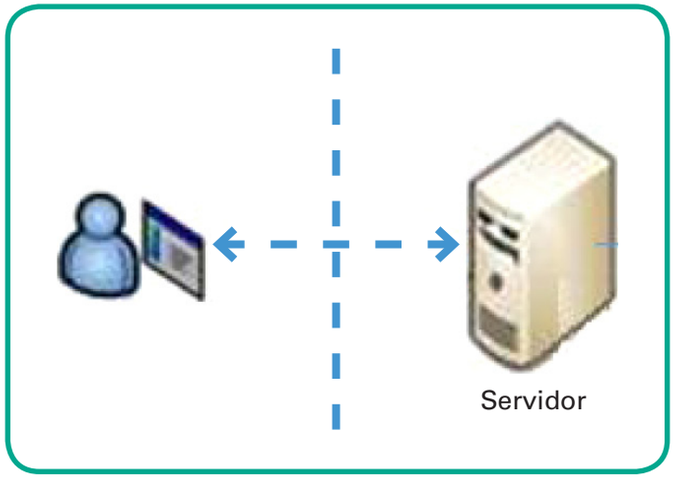


Figura 29: Interfaz de usuario. Fuente: Wong, 2017.

- Interfaces con aplicaciones externas: pueden ser mediante base de datos, a nivel de operación o por el consumo de servicios.

En el ejemplo que se muestra a continuación, se observa la interacción del sistema principal, Sistema de Recursos Humanos, con el servidor de servicios, y un segundo sistema, Sistema de Tesorería. Esta interacción se da a nivel de aplicación y a nivel de base de datos.

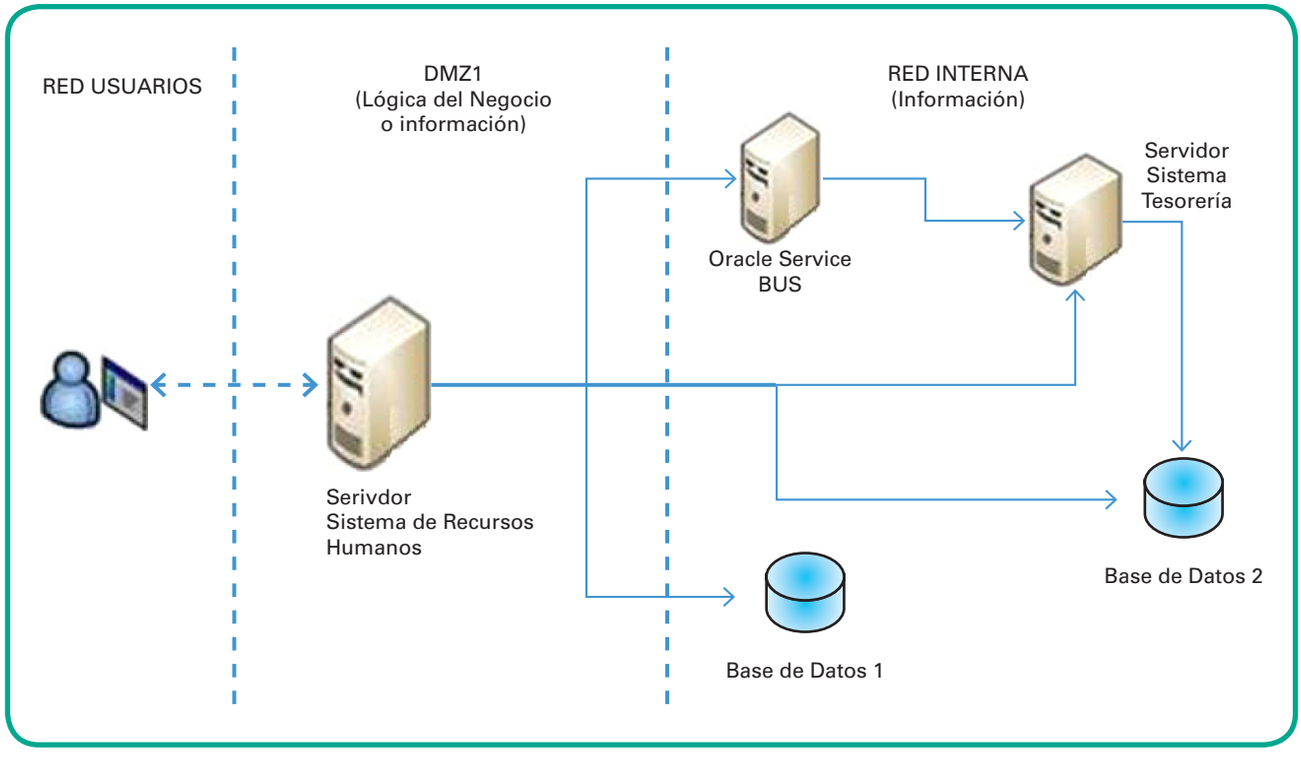


Figura 30: Interfaces de aplicaciones externas. Fuente: Wong, 2017.

- Interfaces con dispositivos de hardware externos: es importante identificar esta interfaz, ya que se requieren efectuar las configuraciones adecuadas para la correcta operación del sistema. Asimismo, es importante contar con este hardware durante el desarrollo para poder efectuar las pruebas necesarias antes del pase al área de control de calidad.

Este análisis permite conocer los límites de la aplicación por interfaces, de modo que se puede definir el alcance a nivel de datos y servicios, así como su funcionalidad. También, se garantiza el nivel de interoperabilidad del sistema.

En el siguiente ejemplo se muestran interfaces a nivel de hardware, en donde vemos la interacción con el internet, la red y los ruteadores. Dependiendo del tipo de sistema, se pueden consignar otros tipos de hardware tales como impresoras, lectoras de códigos de barras, entre otros.

Las interfaces deben estar identificadas desde las etapas tempranas y simular el ambiente en desarrollo; se deben definir las configuraciones y establecer estándares para el pase a producción de los aplicativos. Adicionalmente, al software generado también se deben consignar las configuraciones en el hardware complementario. En estos casos se sugiere efectuar ensayos de esfuerzo que prueben no tan solo el software sino también el hardware. Las pruebas deben ser integrales.

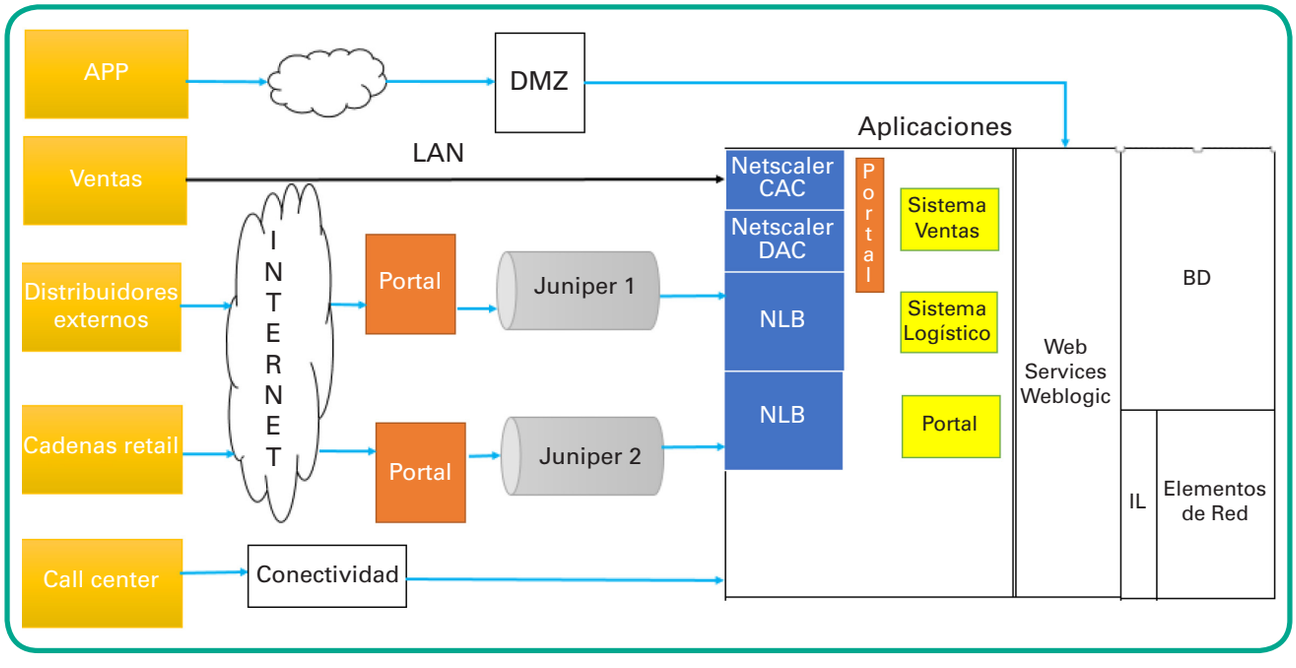


Figura 31: Interfaces con dispositivos de hardware. Fuente: Wong, 2017.

5. Entrevistas

En una entrevista, el entrevistador dirige formal o informalmente con preguntas a una parte interesada; la finalidad es obtener respuestas que se utilizarán para crear requisitos formales. Las preguntas pueden ser estructuradas previamente o de manera espontánea. En este último caso se deberá contemplar la capacidad y experiencia del entrevistador. Las entrevistas pueden realizarse de manera colectiva a un grupo de interesados; sin embargo, las entrevistas individuales son más comunes.

La característica fundamental de las entrevistas es que son cara a cara, se tiene de forma presencial al entrevistado, lo que permite formular varias preguntas hasta cerrar el requisito.

Las entrevistas se catalogan según lo siguiente:

- Entrevista estructurada: la persona que entrevista ya tiene una lista de preguntas predefinidas y busca obtener respuestas que satisfagan las consultas que finalmente darán origen a los requisitos. Espontáneamente y producto de las respuestas pueden surgir preguntas aleatorias; sin embargo, la sesión en general tiene un objetivo y se direcciona en ese sentido.

- Entrevista sin estructurar: el entrevistador y el entrevistado discuten algún tema de manera abierta. Durante el proceso de entrevista se irán generando preguntas en relación con el tema. En este tipo de entrevistas, a fin de dirigir la obtención de requisitos, el entrevistador debe poseer un mayor grado de conocimiento y experiencia en el manejo de entrevistas, de modo que pueda guiar de manera natural la conversación.

La entrevista exitosa depende de varios factores:

- Dominio y experiencia del tema en discusión por parte de la persona que entrevistará.

- Habilidad del entrevistador para conducir la sesión de entrevista y obtener la información necesaria; si el usuario se desvía de la conversación, el entrevistador tiene que regresar al tema inicial.

- Documentar la entrevista y capturar la información relevante de los requisitos. Esta actividad puede ser ejecutada por el entrevistador o una persona externa, que puede ser un observador.

- Conocimiento del entrevistado para proporcionar la información correcta acerca de los requisitos.

6. Observación

Esta actividad implica que una persona observe a otra ejecutar sus actividades en el lugar de trabajo; el objetivo es conocer las actividades que realiza, identificar el proceso y capturar requisitos.

El IIBA denomina a la observación como "sombra de trabajo" o "seguir a la gente", mientras que el PMI lo denomina por sus siglas en inglés *Job shadowning*, que quiere decir "observación de profesionales". Lo enriquecedor de este proceso es observar a una persona experta, plasmar gráficamente el proceso actual y proponer mejoras, las mismas que deben ser validadas previamente por los usuarios finales.

La observación contempla lo siguiente:

- Pasivo/invisible: el observador mira al usuario trabajar en su ambiente de trabajo de inicio a fin, solo observa sin hacer preguntas y registra los datos del proceso. Las preguntas las efectúa al final del proceso. Cabe mencionar que esta actividad de observación puede darse con más de un usuario final; posiblemente el proceso atraviese varias unidades de negocio con diferentes actores. En ese caso será necesario que el observador visualice a diferentes usuarios en diversos momentos y en varias oportunidades, con la finalidad de capturar la información necesaria para el diseño del proceso.

- Activo/visible: el observador visualiza las actividades del usuario y registra los datos del proceso; si algo no le quedó claro puede preguntar en cualquier momento, inclusive paralizando las actividades del usuario.

En algunos casos el observador puede ser parte de la organización o es una persona experta en el negocio. Asimismo, hay personas que no solo son observadores sino que se involucran con el proceso participando activamente en su ejecución para entenderlo mejor.

7. Prototipos

Mediante los prototipos se obtiene la retroalimentación de los requisitos en cuanto a las interfaces; los analistas generan prototipos previos al desarrollo para que el usuario final pueda obtener una experiencia previa a la construcción y, de esta forma, puedan acentuar los requisitos en relación con el software requerido. La elaboración del prototipo implica el término iterativo, pues el prototipo inicial se retroalimenta y genera nuevas versiones hasta que el usuario apruebe el prototipo y pueda pasar a la fase de diseño y construcción.

Con la evolución tecnológica y el manejo de redes sociales, la experiencia a usuario ha cobrado mayor relevancia, por lo que el desarrollo de prototipos para evaluar interfaces y navegación a nivel web y móvil hoy en día constituye una actividad más dentro del ciclo de vida del desarrollo de software.

En la figura 32 se aprecia, a modo de ejemplo, la interfaz móvil y las opciones de prototipo que podrían aplicarse a nivel móvil; de esta manera se presentan los prototipos primero a un nivel monocromático y luego de acuerdo con los estándares de la organización se adaptan los colores según los manuales de identidad, en donde se reflejan lo colores institucionales, tipos de letra, tamaño, entre otros. Aquí se definen los estándares de manera visual para los productos de software de la organización. Se definen colores y opciones de elementos dentro del prototipo de interfaz.

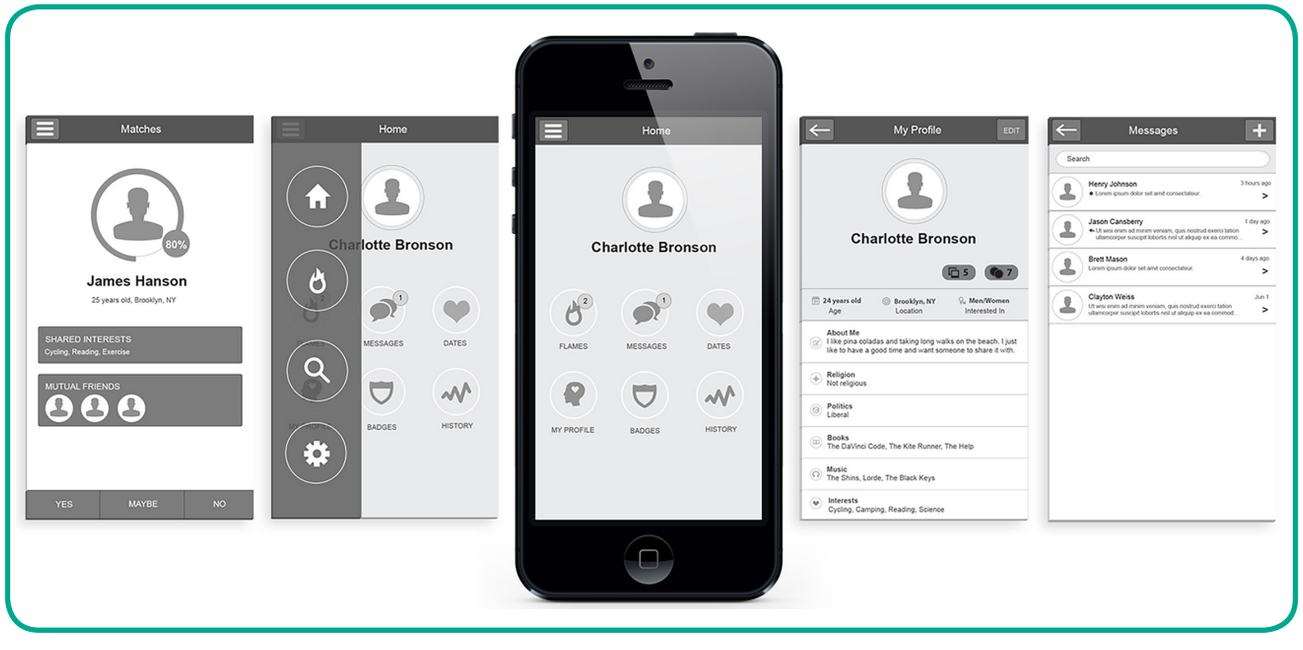


Figura 32: Ejemplo de prototipo. Disponible en http://bit.ly/2xokFCt

En la figura 33 se aprecia la navegabilidad de la aplicación móvil del prototipo; vemos un primer bosquejo de la navegabilidad para que el usuario pueda determinar la secuencia correcta. Esta navegabilidad puede ser mostrada de manera gráfica o mediante un prototipo de alto nivel presentado en móviles.

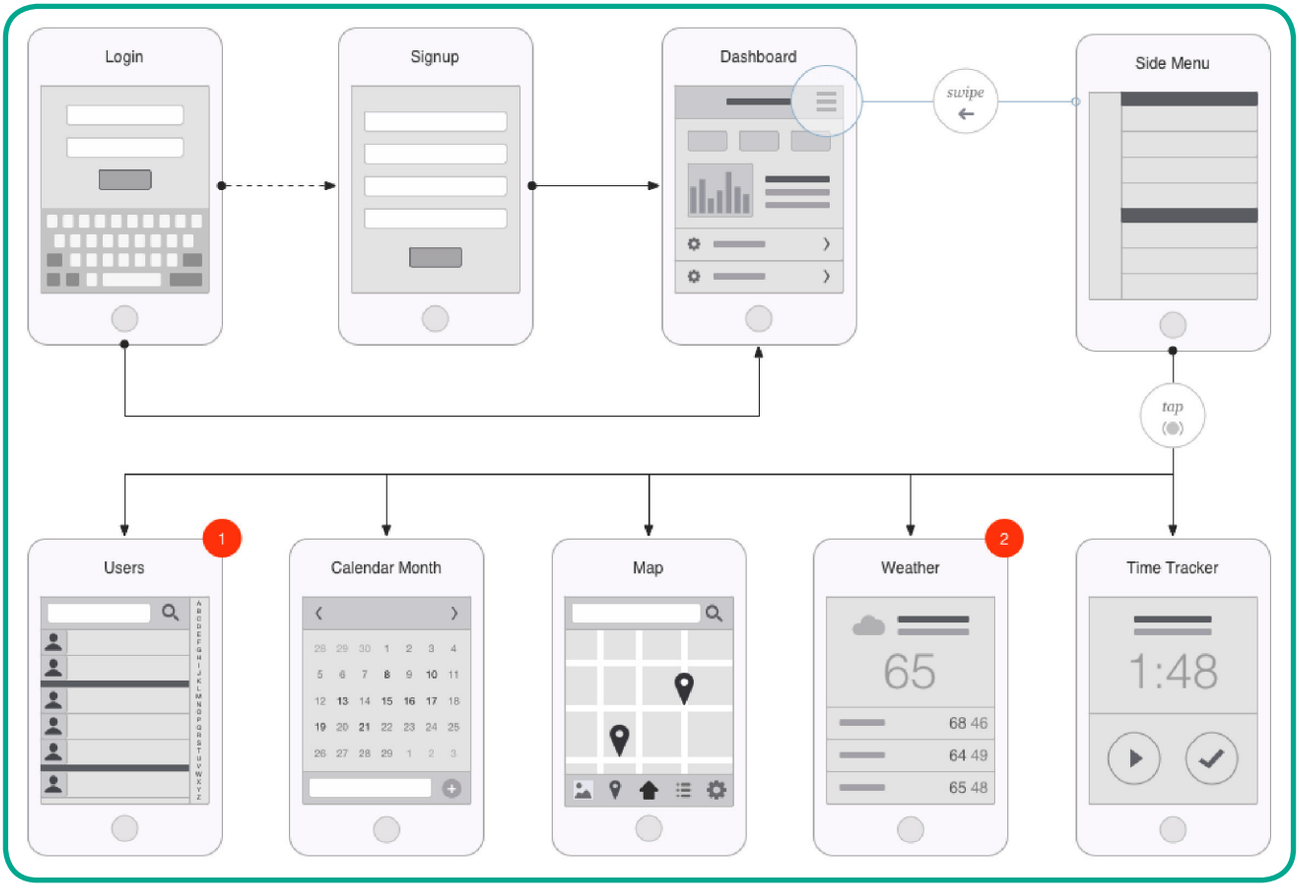


Figura 33: Ejemplo de navegabilidad de prototipo. Disponible en http://bit.ly/2y1egiF

8. Talleres de requisitos

Un taller puede utilizarse para determinar, definir, establecer prioridades y alcanzar el cierre de los requisitos para el desarrollo del software requerido.

Una buena dirección de un taller de requisitos redundará en una eficaz entrega de requisitos de alta calidad. La finalidad es promover un ambiente de confianza, comprensión y comunicación eficaz entre los interesados del proyecto y producir resultados que soporten el análisis y diseño del producto de software.

Un taller de requisitos se enfoca en la atención de usuarios claves y expertos que han sido seleccionados previamente; se da durante un período corto e intensivo (uno o varios días).

El taller puede ser dirigido por un miembro del equipo o por un facilitador experimentado y neutral. Un escriba (también conocido como un registrador) documenta los requisitos suscitados, así como cualquier problema pendiente.

Un taller puede ser utilizado para generar ideas para nuevas características o productos, para llegar a un consenso sobre un tema, o para revisar los requisitos. Otros resultados suelen ser requisitos detallados capturados en modelos.

9. Encuestas / Cuestionarios

Los cuestionarios o encuestas contienen un grupo de preguntas orientadas a obtener información específica de los usuarios rápidamente. Generalmente son solicitadas de manera anónima con la finalidad de obtener información de clientes, productos, servicios, procesos, actitudes, entre otros.

Las respuestas de los cuestionarios o encuestas pueden catalogarse de manera cualitativa o cuantitativa, dependiendo del enfoque de las preguntas.

Las respuestas son analizadas y presentadas para la toma de decisiones requeridas por los interesados que solicitaron la información. Según esta, se tomarán decisiones en relación con los requisitos.

Las preguntas de las encuestas pueden ser:

- Cerradas: el encuestado puede seleccionar la respuesta de un grupo de alternativas; el tipo de encuesta es objetiva con preguntas y respuestas predefinidas. La evaluación de este tipo de encuestas es mucho más rápida y suele ser cuantitativa.

  En el siguiente ejemplo (ver figura 34) se muestra al principio la leyenda de puntajes que cada usuario deberá colocar como parte de las respuestas de las preguntas que posteriormente se efectúan. Cada respuesta deberá tener un número, el mismo que posteriormente será analizado de manera cuantitativa.

LEYENDA DE PUNTAJES			
1. Deficiente	**2. Regular**	**3. Bueno**	**4. Excelente**

I. SOBRE EL CURSO

	Puntaje
1. Los temas desarrollados ¿fueron?	
2. La Metodología empleada (trabajos, resolución de casos, exámenes, etc) ¿fue?	
3. La Duración de la Capacitación ha sido	

II. SOBRE LOS DOCENTES

		Puntaje
Docentes 1	1. El dominio del Docente mostrado sobre el tema ¿fue?	
	2. ¿Como catalogaría Ud. la forma de expresarse del docente?	
	3. El manejo en la conducción de la capacitación (fomenta participación, domina situaciones imprevistas, etc) ha sido	

Figura 34: Ejemplo de preguntas cerradas. Fuente: Wong, 2017.

- Abiertas: El encuestado puede responder en sus propias palabras a la pregunta. Este tipo de encuestas dan libertad de expresión al encuestado, y en algunos casos permite tener una información más detallada. La información de estas encuestas es más difícil de clasificar, pues suelen ser cualitativas.

  En el siguiente ejemplo se muestra un grupo de preguntas abiertas, para que el usuario en cada caso pueda responder lo que considere apropiado desde su perspectiva.

Nro	PREGUNTA	RESPUESTA
1	¿Cómo calificaría el compromiso del proveedor ante la solicitud de alguna atención?	
2	¿El Requerimiento pasó a Producción en la oportunidad requerida?	
3	¿Cuándo se presenta un problema, el proveedor muestra interés en solucionarlo?	
4	¿El proveedor cumplió con las normas, indicaciones y procedimientos en la atención de sus entregables?	
5	¿Cómo calificaría el número de iteraciones que ocurrieron en la atención de sus entregables?	
6	¿El proveedor cumple con los compromisos y/o plazos establecidos?	
7	¿Los tiempos asignados a las actividades de cronograma fueron los adecuados?	
8	¿Cómo calificaría la calidad del entregable de software?	
9	¿Cómo calificaría el nivel de Comunicaciones del responsable del proveedor?	
10	¿El proveedor escucha comprende sus necesidades y participa activamente en la solución?	
11	¿Qué le pareció el trato del personal que atendió su desarrollo de software?	

Figura 35: Ejemplo de preguntas abiertas. Fuente: Wong, 2017.

El PMI sugiere otros tipos de herramientas que pueden apoyar la recopilación de requisitos, tales como las técnicas grupales de toma de decisiones y los estudios comparativos.

10. Técnicas grupales de toma de decisiones

En estas técnicas se manejan múltiples alternativas, se clasifican y asignan prioridades a los requisitos del producto sobre la base de decisiones grupales.

La decisión grupal para agilizar su proceso utiliza lo siguiente:

- Unanimidad: el grupo está de acuerdo y toma una única decisión en cuanto a los requisitos.
- Mayoría: se toma la decisión cuando más del 50% está de acuerdo.
- Pluralidad: refleja la decisión de la mayoría, busca acotar, sobre todo cuando hay más de dos alternativas propuestas.
- Dictadura: solo una persona del grupo toma la decisión.

11. Estudios comparativos

Compara los procesos, prácticas organizacionales, operaciones con otras organizaciones que ejecutan actividades similares. El objetivo de estos estudios comparativos es extraer las mejores prácticas y plasmarlas en requisitos para mejorar.

El resultado de todos estos métodos y herramientas son los requisitos que deberán estar alineados con los objetivos estratégicos de la organización. Los requisitos pueden ser expresados en principio a alto nivel y gradualmente pueden ser requisitos más detallados. Los requisitos deben de ser concretos y no ambiguos, pues de no estar bien definidos, la etapa de análisis puede presentar desviaciones en plazo o el producto puede no cumplir con las expectativas del cliente.

Lectura seleccionada n.º 4

Mcdonald Landázuri, B. (2005). *Definición de perfiles en herramientas de gestión de requisitos* (Programa de Doctorado). Disponible en http://bit.ly/2gijIWI

Actividad n.º 4

Foro de discusión sobre fundamentos de análisis.

Instrucciones:

- Ingrese al foro y participe con comentarios críticos y analíticos del tema modelos de sistemas.
- Lea y analice el tema n.º 3 del manual.
- Responda en el foro a las preguntas acerca del modelado de sistemas:

  Mencione tres técnicas de elicitación distintas a las señaladas en el manual autoformativo.

Tarea académica n.° 2

I. Trabajo de investigación

a. Hacer un *paper* sobre elicitación de requerimientos.
   i. El *paper* debe tener una estructura correcta y las fuentes deben referenciarse según el formato APA vigente.
   ii. Los puntos mínimos que se deben explicar en el *paper* son los siguientes:
      1. Descripción (caso de negocio).
      2. Aplicación de cuatro técnicas de elicitación de requerimientos.
         - Descripción de la técnica.
         - Información obtenida producto de la ejecución de la técnica.
         - Herramientas utilizadas o estrategias.
      3. Lista de requisitos funcionales y no funcionales.
      4. Conclusiones y recomendaciones.

b. Una presentación sobre el tema que le fue indicado en el punto anterior.
   i. Resaltando los puntos importantes del tema.
   ii. No copiar y pegar texto para explicar el tema.
   iii. Podrá usar imágenes y escribir palabras que considere importantes.

c. Un video subido a Youtube donde se explique su diapositiva para que sea luego compartida en clase.

II. Rúbrica

n.°	Rúbrica	Descripción	Min. puntaje	Máx. puntaje
1	Documento	El trabajo escrito debe presentar todas las pautas expuestas en formato *paper*.	0	10
2	Presentación	Elaboración de diapositivas para la presentación del trabajo.	0	6
3	Video	Archivo de texto donde se encuentre la URL del video en Youtube en el que se explica la diapositiva presentada.	0	4

III. Indicaciones

- Todos los puntos de la rúbrica deberán ser subidos a la plataforma en formato ZIP y como nombre del archivo deberán tener el siguiente formato: **APELLIDO_NOMBRE.zip**
- En el caso de plagio, el trabajo será invalidado y la nota será de 00.
- No referenciar adecuadamente según la norma APA vigente le restará un punto por cada referencia no establecida.

Glosario de la Unidad II

C

Caso de negocio: Una evaluación de los costos y beneficios asociados con una iniciativa de propuesta (IIBA, 2009).

F

Fiabilidad: Capacidad de un sistema para entregar los servicios como se especifican. La fiabilidad se puede especificar cuantitativamente como la probabilidad de que ocurra un fallo de funcionamiento o como la tasa de ocurrencia de estos (Sommerville, 2011).

I

Interfaz: Especificación de los atributos y operaciones asociados con un componente software. La interfaz es utilizada como el medio de tener acceso a la funcionalidad del componente (Sommerville, 2011).

P

Principios de diseño de las interfaces de usuario: Conjunto de principios que expresan buenas prácticas para el diseño de interfaces de usuario (Sommerville, 2011).

Prototipo: Una versión parcial o preliminar del sistema (IIBA, 2009).

R

Requerimiento funcional: Declaración de alguna función o característica que se debe implementar en un sistema (Sommerville, 2011).

Requerimiento no funcional: Declaración de una restricción o comportamiento que se aplica a un sistema. Esta restricción se puede referir a las propiedades emergentes del software que se está desarrollando o al proceso de desarrollo (Sommerville, 2011).

U

Usuario: Individuo u organización que utiliza el sistema en operación para llevar a cabo una función específica (Indecopi, 2006).

Validación: Confirmación mediante el suministro de evidencia objetiva de que se han cumplido los requerimientos para una utilización o aplicación específica prevista (Indecopi, 2006).

Bibliografía de la Unidad II

Al-Zawahreh, H., & Almakadmeh, K. (2015). Procedural model of requirements elicitation techniques. En Eddine Boubiche, D., Hidoussi, F., & Toral Cruz, H. (eds.), *Proceedings of the International Conference on Intelligent Information Processing, Security and Advanced Communication* (pp. 1-6). Batna, Argelia: ACM.

Bourque, P., & Fairley, R. (2014). *SWEBOK V3.0. Guide to the Software Engineering Body of Knowledge*. New Jersey: IEEE. Disponible en http://bit.ly/2zcSdFX

Fraser, M. D., Kumar, K., & Vaishnavi, V. K. (1991). Informal and formal requirements specification languages: Bridging the gap. *IEEE Transactions on Software Engineering*, *17*(5), 454-466. Disponible en http://bit.ly/2xoNLlu

International Institute of Business Analysis. (2009). *A Guide to the Business Analysis Body of Knowledge (BABOK Guide). Version 2.0*. Ontario (Canadá): autor. Disponible en http://bit.ly/2gwEYrW

Isabirye, N., & Flowerday, S. (2008). A model for eliciting user requirements specific to South African rural areas. En *Proceedings of the 2008 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists on IT research in developing countries: Riding the wave of technology* (pp. 124-130). Wilderness, Sudáfrica: ACM.

Kendall, K., & Kendall, J. (2013). *Systems analysis and design* (9a ed.). New Yersey: Prentice Hall.

Pressman, R. (2005). *Ingeniería del Software. Un enfoque práctico* (6a ed.). D.F. México: McGraw-Hill.

Sommerville, Ian. (2011). *Software Engineering* (9a ed.). Madrid: Pearson Education.

Wiegers, K., & Beatty, J. (2013). *Software Requirements* (3a ed.). Washington: Microsoft Press. Disponible en http://bit.ly/2ghNk6M

Xie, H., Liu, L., & Yang, J. (2009). Optimizing requirements elicitation process based on actor preferences [paper]. En *Proceedings of the 2009 ACM Symposium on Applied Computing*, Honolulu, Hawaii.

Autoevaluación n.º 2

1. En el diseño de un sistema de telefonía celular, ¿cuál de los siguientes corresponde a un requerimiento funcional?
   a) Enviar imágenes en un SMS.
   b) Número de caracteres de un mensaje.
   c) Poder escribir por Facebook.
   d) El teléfono tiene que ser seguro.
   e) Recibir llamadas.

2. Durante el desarrollo de una aplicación, en la etapa de diseño de prototipos se determinó que hubo observaciones en la etapa de definición de requerimientos. Dichos errores deberían:
   a) No ser considerados para correcciones del documento de especificación de requerimientos y simplemente ser atendidos en lo que resta del proceso de desarrollo.
   b) Deberían ser considerados para levantar las observaciones del documento de requerimientos y volver a definirlos de manera correcta.
   c) Debería gestionarse una solicitud y cobrarle más al cliente para corregir los errores.
   d) Estos errores son indiferentes al proceso de desarrollo del software, y no deberían ser tomados en cuenta debido a que ya se tenía una planificación previa a la realización de los prototipos.
   e) Estos errores servirán para mejorarlos y usarlos en el desarrollo de otro software.

3. Al utilizar un modelo de proceso de software para el desarrollo de un sistema lo más difícil que se puede encontrar sería:
   a) La complejidad del sistema por desarrollar.
   b) Implementar el nuevo requerimiento.
   c) Reutilizar código de sistemas antiguos.
   d) Conversar con el cliente.
   e) Aplicar la metodología XP.

4. La creación de prototipos ayuda a:
   a) Diseñar y programar el sistema que se implementará.
   b) La implementación del sistema y a comprobar su operación.
   c) Probar las opciones de diseño y entender mejor el problema y su solución.
   d) Analizar la base de datos del sistema.
   e) Generar la arquitectura del sistema.

5. Se desea realizar un sistema de atención al cliente online para una operadora de telefonía móvil. Se ha determinado como requerimiento que el sistema funcione de acuerdo con el horario laboral de los empleados, es decir, de 18:00 a 19:00 horas. El presente requerimiento no funcional puede ser clasificado como:
   a) Requerimiento del producto.
   b) Requerimiento de la organización.
   c) Requerimiento externo.
   d) Requerimiento de arquitectura.
   e) Requerimiento de portabilidad.

6. Al momento de crear un software, al no tener claros los requerimientos se corre el riesgo de:
   a) Perder tiempo y dinero en el desarrollo del software
   b) Dar una mala reputación de la empresa
   c) No desarrollar un buen sistema para el cliente.
   d) Perder clientes
   e) Generar un software que puede servir para otro cliente

7. En un sistema de reservación de vuelos, ¿cuál de los siguientes requerimientos es de sistemas?
   a) El sistema debe funcionar correctamente en cualquier navegador.
   b) El sistema no debe tardar más de cinco segundos en mostrar los resultados de una búsqueda.
   c) El sistema debe de ser trabajado en lenguaje Java y Base de Datos Orcle 10g.
   d) El sistema debe contener 10 casos de uso como máximo.
   e) El usuario deberá tener la posibilidad de buscar por fecha del vuelo, tipo vuelo, disponibilidad de asientos y costo de los vuelos.

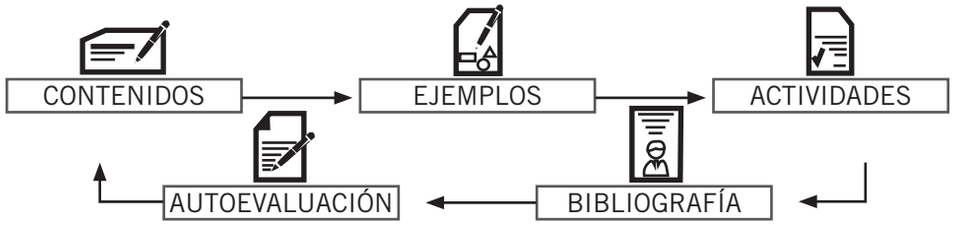
8. En el conocido juego de dados del 7: El jugador tira ambos dados y suma las caras superiores. Previamente, se hace una apuesta sobre lo que será la suma de los dados. En el caso de que haya apostado a que la suma sería abajo del siete y acertó, gana la suma apostada. Lo mismo sucede cuando apuesta arriba del 7. En el caso de que haya elegido el 7 y acierta, gana el triple de la apuesta. En el caso de no acertar, pierde la cantidad apostada. Uno de los requerimientos funcionales pudiera ser:
   a) El jugador podrá elegir las siguientes jugadas: arriba del 7, debajo del 7 y el 7.
   b) El sistema no deberá registrar el capital inicial del jugador.
   c) El sistema no deberá generar los valores de una cara cuando se tire el dado.
   d) El sistema deberá funcionar en todos los navegadores.
   e) El sistema deberá tener interfaces amigable.

9. En el siguiente requerimiento: "A cada pedido se le deberá asignar un identificador único (ID_PEDIDO)" corresponde a un requerimiento:
   a) No funcional
   b) Funcional
   c) No es un requerimiento
   d) De sistema
   e) De interface

10. El análisis de documentos es una técnica de:
    a) Análisis funcional
    b) Análisis no funcional
    c) Análisis de usuario
    d) Análisis de sistemas
    e) Elicitación

UNIDAD III
OPCIONES, FINANCIAMIENTO Y ADMINISTRACIÓN DE RIESGOS

DIAGRAMA DE PRESENTACIÓN DE LA UNIDAD III



ORGANIZACIÓN DE LOS APRENDIZAJES

RESULTADO DE APRENDIZAJE: Al finalizar la unidad, el estudiante será capaz de organizar la especificación de requerimientos de software de su proyecto.

CONOCIMIENTOS	HABILIDADES	ACTITUDES
Tema n.° 1: Documentación de los requerimientos 1. Documentación de ciclo de vida 1.1. Documento de modelo de proceso de negocio 1.2. Documento de análisis 1.3. Documento de diseño 1.4. Manuales de usuario 1.5. Evidencias de pruebas unitarias 1.6. Evidencias de pruebas integrales 1.7. Documento de pase a producción 1.8. Plan de pruebas 1.9. Casos de prueba 1.10. Evidencias de pruebas 1.11. Informe de Calidad **Lectura seleccionada n.° 5** *Notas del curso: Análisis de Requerimientos* (Gómez, 2011). **Tema n.° 2: Técnicas de especificación de requerimientos** 1. Especificación de Requerimientos de Software – IEEE 830 1.1 Naturaleza de la especificación de requerimientos de software 1.2 Medio ambiente de la especificación de software. 1.3 Características de la especificación de requerimientos de software 1.4 Preparación conjunta de la especificación de requerimientos de software 1.5 Evolución de la especificación de requerimientos de software 1.6 Prototipado 1.7 Diseño de inclusión en la especificación de requerimientos de software 1.8 Incorporación de los requisitos del proyecto de especificación de requerimientos de software **Lectura seleccionada n.° 6** *Rational Unified Process. Best Practices for Software Development Teams* (Rational Software, 1998). **Autoevaluación de la Unidad III**	1. Reúne la especificación de requerimientos de software en un documento según el estándar IEEE 830. **Actividad n.° 5** Los estudiantes participan en el foro de discusión sobre análisis de requerimientos. **Tarea académica n.° 3**	1. Coopera en la elaboración del documento de la especificación de requerimientos de software.

Documentación de los requerimientos

Tema n.° 1

En el siguiente capítulo, conocerá los diferentes documentos por considerar como parte de la fase análisis. Asimismo, comprenderá las diferentes técnicas de especificación utilizadas en la obtención de requerimientos de software. Cabe mencionar que la especificación de requerimientos se define posterior a la elicitación de requerimientos, que es el proceso base para la obtención de estos. Cuando ya se cuenta con los requerimientos definidos sean de negocio, partes interesadas, soluciones (requisitos funcionales y no funcionales) y requisitos de transición, se puede dar inicio al modelo de proceso de negocio y, basado en él, elaborar la documentación necesaria para el desarrollo del requerimiento.

1. Documentación del ciclo de vida

La documentación de los requerimientos contempla las especificaciones funcionales y técnicas que servirán para el desarrollo del software. Adicionalmente, el desarrollo luego de su implementación contemplará documentación que servirá como guía para el usuario en la funcionalidad del sistema, así como documentación técnica de configuración e instalación del producto.

Cabe mencionar que la documentación variará de acuerdo con la metodología de desarrollo utilizada, las cuales pueden ser RUP, Xtreme programming, Microsoft solution framework, SCRUM (métodos ágiles), entre otras. Para todos los casos, la documentación más relevante que se utiliza como parte del ciclo de vida incluye lo siguiente:

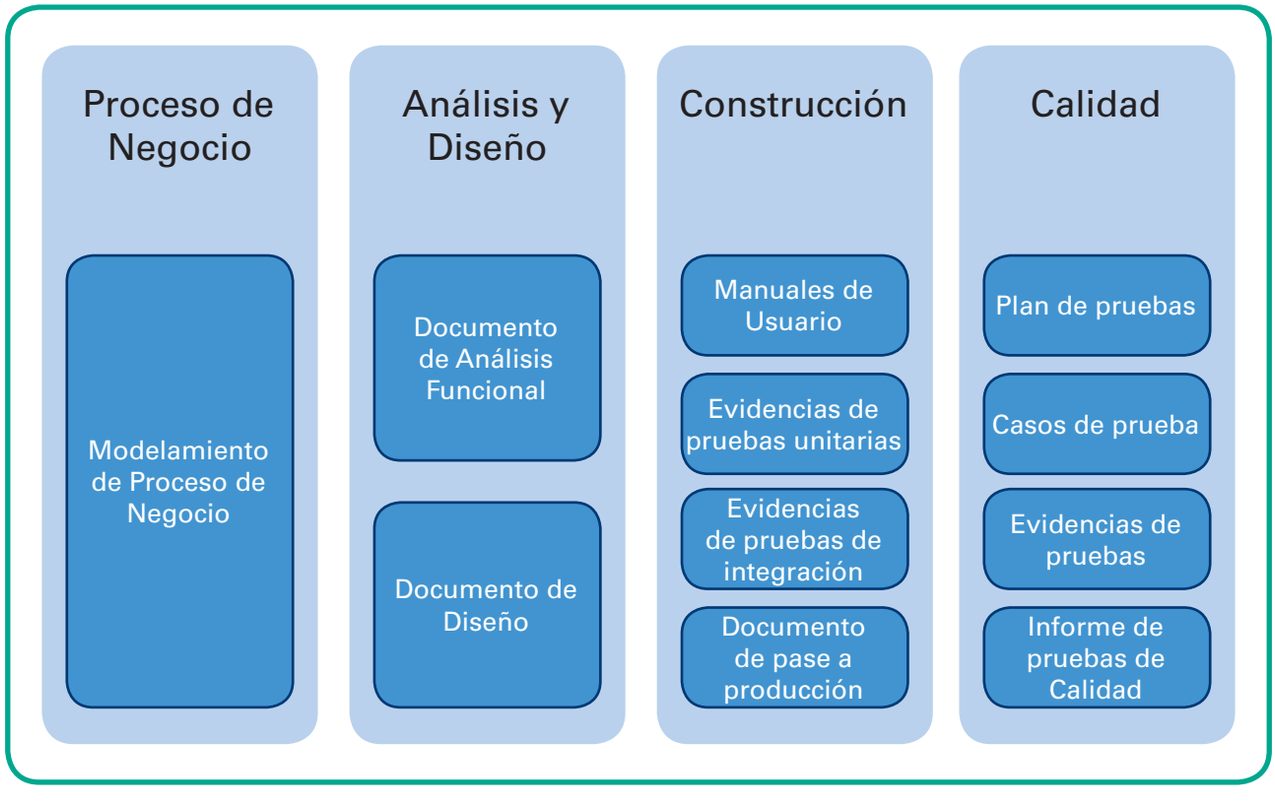


Figura 36: Documentación de requerimientos. Fuente: Wong, 2017.

A continuación se mencionan los documentos por considerar como documentación de los requerimientos.

1.1 Documento Modelo de proceso de negocio

El modelo de proceso de negocio otorga una vista general del negocio del requerimiento por desarrollar en términos funcionales, la misma que puede generarse mediante un diagrama gráfico, como el diagrama de contexto y el diagrama de flujo de datos.

Existen herramientas que automatizan el flujo desde el inicio del negocio e implementan desde esta fase un BPMN (Business Process Model and Notation), que es la notación del modelado de proceso de negocio. Este modelado puede desarrollarse. Entre las herramientas más conocidas se encuentran las tecnologías IBM y Oracle, que permiten agilizar las implementaciones del software

Veamos cuáles son los ítems necesarios que considerará este documento.

1.1.1 Modelamiento de la situación actual

- **Descripción del proceso actual**

  En el caso de un proceso de negocio nuevo, no se requiere modelamiento de situación actual. Para el caso de modificación de un proceso existente, se debe ubicar el proceso dentro de un mapa de macroprocesos de la organización a fin de poder ubicar en qué parte del proceso se efectuará la modificación.

- **Modelamiento del proceso actual**

  Deberá incluir lo siguiente:

  - Diagrama de procesos: muestra a detalle el flujo de trabajo actual. Asimismo, denota los roles que intervienen.
  - Detalle de las actividades: se deben definir las actividades de entrada y salida, el detalle del proceso y roles que intervienen.
  - Reglas de negocio: se describen las reglas de negocio actuales.

- **Análisis cualitativo y cuantitativo**

  Se considera información estadística que servirá para el monitoreo del comportamiento del sistema. Esta información debe considerarse obligatoria cuando existe procesamiento de información como parte del desarrollo del requerimiento de software.

1.1.2 Modelamiento de la situación propuesta

- **Descripción del proceso propuesto**

  En el caso de un proceso de negocio nuevo, se requiere un modelamiento completo. En el caso de modificación de actividades de un proceso de negocio, se requiere actualizar el diagrama de proceso actual y especificar en detalle las reglas de negocio.

- **Modelamiento del proceso propuesto**

  Deberá incluir lo siguiente:

  - Diagrama de procesos: muestra a detalle el flujo de trabajo propuesto. Asimismo, denota los roles que intervienen.

- Detalle de las actividades: se deben definir las actividades de entrada y salida, el detalle del proceso y roles que intervienen.
- Reglas de negocio: se describen las reglas de negocio propuestas para el desarrollo del requerimiento de software.

- **Análisis cualitativo y cuantitativo**

  Se deberá especificar el comportamiento del requerimiento de software una vez establecido en relación con la información proporcionada.

  Por ejemplo:

  - Si el proceso manualmente demoraba 10 días, con la implementación del software demorará 4 días.
  - El procesamiento de información de 3000 datos demora actualmente siete días; con el sistema, el mismo procesamiento demorará un día.

1.1.3 Requerimientos del proyecto

- **Requerimientos de normas legales**

  Se describirán las normas legales en el caso de que impacten en el desarrollo del requerimiento de software.

- **Requerimientos de bienes y servicios**

  Si el desarrollo de software involucra la adquisición de nuevo hardware o software, los mismos deberán ser descritos.

- **Requerimientos de comunicación y difusión**

  Se describirá el proceso de despliegue del software a nivel interno en la organización, así como a nivel externo.

1.2 Documento de análisis

El documento de análisis contempla la especificación detallada de los requerimientos de software, los mismos que cubren las necesidades del usuario. Estos se describen como reglas de negocio y constituyen la descripción inicial del sistema. Este documento se caracteriza por contemplar en términos funcionales los requisitos del usuario, los mismos que al final de su elaboración deberán ser validados.

1.2.1 Modelamiento de requerimientos

- **Alcance**

  En esta actividad se delimita el alcance del sistema, se elabora inicialmente el modelo de procesos propuesto especificado en el modelo de procesos de negocio; se describen los procesos que formarán parte del sistema y se identifican las entidades externas al sistema que aportarán o recibirán datos. Se elabora lo siguiente:

  - Diagrama de contexto del sistema
  - Modelamiento de procesos de negocio.

- **Obtención y análisis de requerimientos**

  En esta fase se describen los requerimientos que debe cumplir el sistema. Asimismo, se definen las prioridades de los requisitos, considerando los criterios de los usuarios acerca de las funcionalidades por cubrir. Los principales tipos de requerimientos que se deben especificar son:

  - Requerimientos funcionales
  - Requerimientos no funcionales
  - Requerimientos de rendimiento
  - Requerimientos de seguridad
  - Requerimientos de implantación
  - Requerimientos de disponibilidad

  En la siguiente tabla se muestra un ejemplo de lista de requisitos funcionales, su detalle y la implementación de regla de negocio relacionada con el requisito funcional. Cabe mencionar que es factible que un requisito pueda tener más de una regla de negocio, por lo que será necesario describirlas todas. Esta información servirá de base para la elaboración de los diagramas que se desarrollarán posteriormente a nivel técnico para la generación del sistema.

  En el caso de que se requiera efectuar una trazabilidad desde el modelo de negocio, se deberá agregar una columna referenciando el código de requisito del modelo de negocio. La trazabilidad de requisitos es importante para no perder de vista algún requisito de usuario mapeado en el modelo de negocio, por lo que el analista debe asegurarse de considerar adecuadamente todos los requisitos.

Tabla 7
Ejemplo de requisitos funcionales y reglas de negocio

N°	Requisito Funcional	Detalle	Regla de negocio
01	Implementar el módulo de planillas	El módulo deberá generar lo siguiente: o Tipo de planilla o Selección de empleados	El sistema listará sólo los empleados pertenecientes al tipo de planilla para la selección.

Nota: Tomado de Wong, 2017.

En el siguiente cuadro se listan *requisitos no funcionales*. En este caso se describen por funcionalidad, fiabilidad y usabilidad; asimismo, se precisa el impacto que ocasionaría en la aplicación por su nivel de criticidad, donde 1 es el más leve y 4 el más severo.

Tabla 8
Ejemplo de lista de requisitos no funcionales

Nº	Nombre del Requisito No Funcional	Descripción detallada	Impacto (1-4)
1	**Funcionalidad**		
1.1	Seguridad	Los usuarios deben tener acceso a las opciones necesarias para el cumplimiento de sus funciones	4
1.2	Auditoria	El sistema deberá almacenar como mínimo la siguiente información: - Fecha y hora de registro - Fecha y hora de modificación - Usuario de registro - Usuario de Modificación	4
2	**Fiabilidad**		
2.1	Disponibilidad	El sistema deberá soportar una disponibilidad de sus servidores de 24x7 al 98% de disponibilidad.	3
3	**Usabilidad**		
3.1	Claridad y no ambigüedad	Los textos y mensajes deberán ser claros, sin ambigüedades de tal manera que sean entendibles para un usuario con un conocimiento básico en el uso de la computadora.	3
3.2	Paginación para número de filas por resultado de búsqueda.	Todas las grillas que provienen de una consulta, de un registro deben tener como máximo una paginación de 10 registros por página, de tener más deben presentar la navegación de páginas.	3
3.3	Datos obligatorios	Todos los campos que son obligatorios, deben tener su respectiva etiqueta con letra negrita.	3

Fuente: Wong, 2017

- **Obtención del Modelo de casos de uso**

  Sobre la base de los requerimientos funcionales se generan los casos de uso del sistema. Debe existir una correspondencia entre ambos.

  La finalidad de esta actividad es especificar cada caso de uso desarrollando los escenarios, y describiendo los flujos principales y los alternativos. Se deberá generar como parte de esta actividad lo siguiente:

  - Diagramas de casos de uso
  - Diagrama de actividades
  - Prototipos del sistema

- **Descripción de perfiles**

  La finalidad de esta actividad es identificar todos los perfiles de usuario, según su nivel de responsabilidad y rol que ejecuta dentro del proceso. Asimismo, se deberán analizar las características más relevantes de los usuarios que van a asumir como actores y su interacción con el sistema por desarrollar. Se deberá generar como parte de esta actividad lo siguiente:

- Lista de perfiles y roles en el sistema

En la siguiente tabla se muestra un ejemplo de registro de perfiles, se detalla el perfil y tipo de acceso.

Tabla 9
Ejemplo de descripción de perfiles

Perfil	P03 - Usuario de Tesorería
Opciones a las que tiene acceso el perfil	Tipo de Acceso
Aprobar / Observar Liquidación	Lectura - Escritura

Nota: Tomado de Wong, 2017.

- **Especificación del comportamiento dinámico de la interface**

Se definen los flujos entre las distintas interfaces de pantalla y se desarrolla una propuesta de navegación. La misma puede ser de manera gráfica a nivel de maquetado o con codificación de alto nivel.

Para cada pantalla se identifica la entrada lógica de los datos y reglas de validación.

El comportamiento está dirigido y representado por los controles y los eventos que provocan su activación. Se deberá generar como parte de esta actividad lo siguiente:

- Diseño de prototipos
- Diagrama de transición de estados
- Diagrama de interacción de objetos.

A continuación se muestra un ejemplo de prototipo. Existen muchas herramientas de generación de prototipos disponibles en el mercado. Para efectos del ejemplo se trabajó con herramienta mockups.

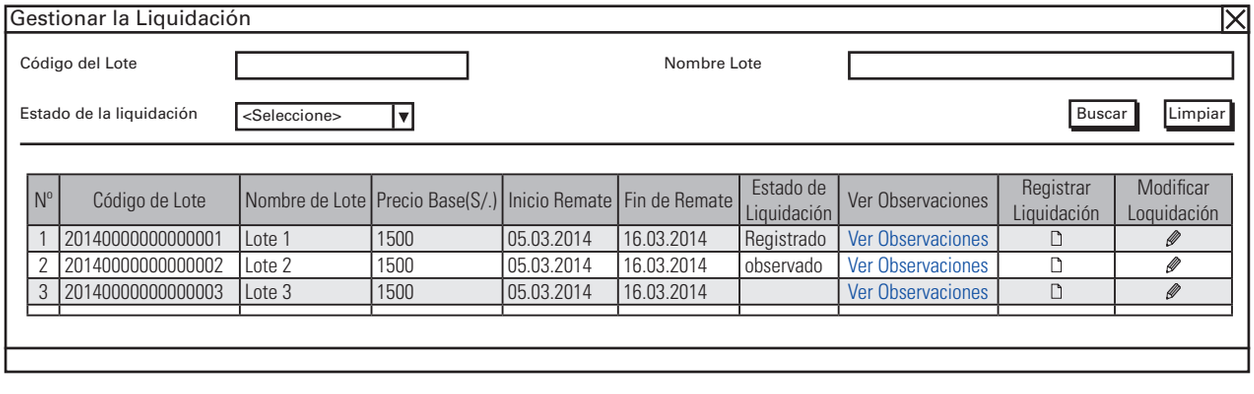


Figura 37: Ejemplo de prototipo. Fuente: Wong, 2017.

- **Formatos de impresión o digitalización**

Se especifican los formatos y características de páginas requeridas como salidas o entradas impresas del sistema. Asimismo, se precisa si se va a considerar información digitalizada. Esta información es necesaria, ya que si se trata de un sistema que almacenará este

tipo de información, se debe tener precaución con el espacio de base de datos. En estos casos, las estadísticas de volumen de datos deberán ser mucho más exhaustivas y estos requisitos deben estar contemplados desde la fase inicial del proyecto con un seguimiento continuo, porque de lo contrario, durante la operación del software en ambiente de producción pueden generarse problemas de almacenamiento y, como consecuencia, una baja en la operación del software.

1.2.2 Análisis de casos de uso

Se identifican los casos de uso y los flujos principales y alternativos. Se describe cada caso de uso precisando la descripción del CUS, el flujo, los actores, las precondiciones, poscondiciones y anotaciones adicionales. Se deberá generar como parte de esta actividad lo siguiente:

- Diagrama de casos de uso
- Descripción de caso de uso

A continuación, se muestra un ejemplo de caso de uso y su descripción. Cabe mencionar que cada CU deberá tener su correspondiente descripción: El ejemplo muestra un flujo con 3 casos de uso.

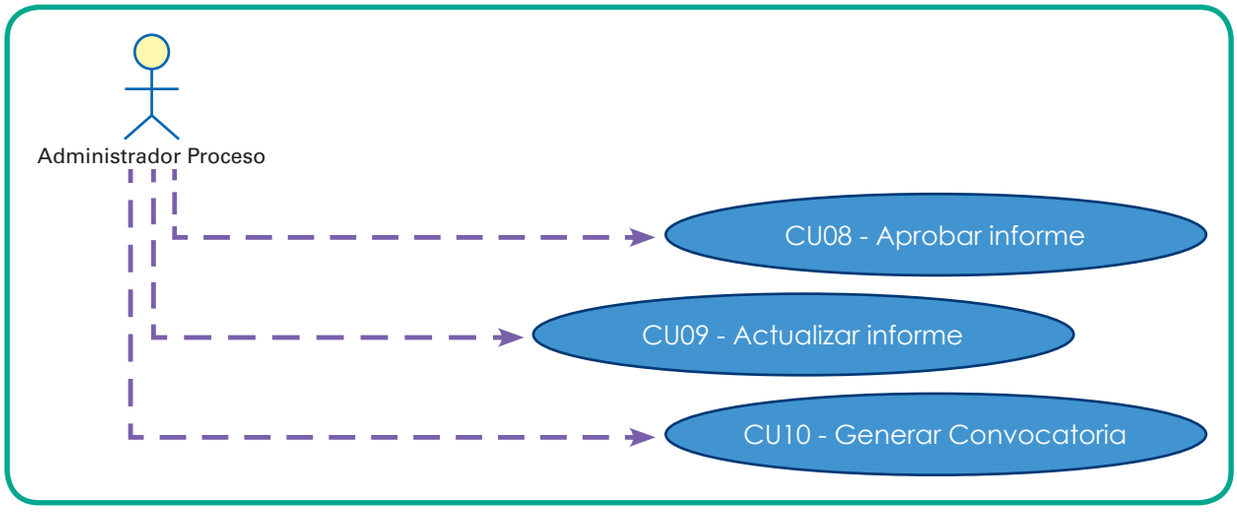


Figura 38: Caso de uso. Fuente: Wong, 2017.

Todos los casos de uso deben tener su correspondiente descripción de caso de uso. En la figura anterior se visualizan tres casos de uso; entonces deberán tener tres descripciones, una por cada caso de uso. A continuación se muestra la descripción de uno de sus casos de uso; para el ejemplo se considera el nombre, la descripción, los actores, precondiciones, poscondiciones, flujo de eventos y, opcionalmente, flujo alternativo, excepciones. Adicionalmente, se contempla el requisito funcional asociado y el prototipo de interfaz.

Tabla 10
Ejemplo de descripción de caso de uso

1. Nombre del Caso de Uso del Sistema		CU08 - Aprobar Informe
2. Descripción del Caso de Uso		
Permite aprobar informes		
3. Actor(es)		
Administrador de Proceso		
4. Precondiciones		
Existencia de informes		
5. Post - condiciones		
Informes aprobados		
6. Flujo de eventos *		
Nro.	Acción de Actor	Respuesta del Sistema
1	El usuario inicia el proceso de Aprobación del Informe	El sistema realizará lo detallado en el RF7
7. Flujo alternativo		
Nro.	Acción de Actor	Respuesta del Sistema
1		
8. Excepciones		
Nro.	Descripción	
9. Requisito funcional asociado		
RF7		
10. Prototipo de interfaz de usuario		
I17		

Nota: Tomado de Wong, 2017.

1.2.3 Análisis de clases

Se describe cada una de las clases identificadas, se describen las responsabilidades que tienen asociadas, sus atributos, y relaciones entre ellas. Este análisis debe contemplar la revisión de las *librerías de clases* existentes para maximizar la reutilización. Se deben considerar los estándares definidos de modo que exista trazabilidad con lo desarrollado en el diseño y construcción. Se deberá generar como parte de esta actividad lo siguiente:

- Diagrama de clases

1.2.4 Análisis de paquetes

Se representa la integración de los módulos del sistema mediante paquetes de clases, que incluyen a los que identifican servicios comunes a varios módulos. Este análisis debe contemplar la revisión de las *librerías de paquetes de clases* existentes para maximizar la reusabilidad.

1.2.5 Elaboración del modelo de datos

Se elabora el modelo de datos que considera todas las entidades, relaciones, atributos y reglas de negocio.

A partir del modelo conceptual, se van incorporando entidades que van apareciendo como resultado del desarrollo del análisis de los requisitos de las necesidades de información del usuario. Luego se genera el modelo lógico de datos y se asegura la normalización para obtener el modelo lógico de datos normalizado.

1.2.6 Especificación de necesidades de migración

Esta actividad se realiza si parte de la funcionalidad desarrollada requiere información de otros sistemas ya existentes, o si es necesario efectuar una carga inicial de información; para ello se toma como base el modelo lógico.

1.3 Documento de diseño

En este documento se definen lo componentes técnicos de la aplicación, se define la arquitectura, el entorno tecnológico y la especificación de los componentes.

1.3.1 Definición de la arquitectura

Se define la arquitectura del sistema, se especifican las particiones físicas, la descomposición lógica en subsistemas de diseño y la ubicación de cada subsistema en cada partición. Se debe precisar la infraestructura necesaria para la ejecución del sistema, y en el particionamiento físico del sistema se identifican los nodos y las comunicaciones.

La división de subsistemas se basa en lo establecido en el documento de análisis. Se debe considerar la facilidad de un mantenimiento de la aplicación en el tiempo, identificación de la funcionalidad común, la reutilización de componentes del sistema, entre otros.
En los subsistemas se debe especificar el diseño de los módulos y clases que forman parte de cada uno de ellos. Se debe considerar el nivel de reutilización de los subsistemas y sus servicios. Se recomienda seguir las siguientes pautas:

- Se debe verificar si es necesario una descomposición de los subsistemas en servicios, entendiendo como tales a módulos o clases independientes, de modo que puedan ser reutilizables.
- Se debe realizar una descripción de la interface y del comportamiento de cada servicio a detalle. Asimismo, efectuar el registro del mismo en la lista de servicios de la organización, de modo que si otro sistema lo requiere pueda invocarlo o reutilizarlo.
- La especificación y diseño de cada servicio, módulo o clase, se realiza con las técnicas habituales de especificación y diseño de módulos o clases (UML).
- A medida que se lleva a cabo esta tarea pueden surgir comportamientos de excepción que deberán contemplarse en el diseño, y que en función del nivel de especificación que se haya establecido, se incorporan a la lista de excepciones.

Los subsistemas identificados se catalogan como específicos o genéricos, asignando cada subsistema a su nodo correspondiente. Los subsistemas específicos contemplan las funcionalidades propias del sistema, mientras que los subsistemas genéricos cubren servicios comunes, proporcionando un acceso a los distintos recursos.

Los subsistemas genéricos contemplan lo siguiente:

- Comunicación entre subsistemas
- Gestión de datos (acceso a datos)
- Ejecución de bacheros (sistema por lotes), migración de datos, sincronización de bases de datos.

- Manejo de transacciones
- Control y gestión de errores
- Seguridad, gestión de accesos

También se listan las excepciones del sistema, las mismas que deberán considerar lo siguiente:

- Descripción de la excepción
- Precondiciones del sistema
- Componente afectado (nodo, módulo, caso de uso)
- Respuesta del sistema
- Componente asociado a la respuesta esperada del sistema (módulo, clase, procedimiento, entre otros)

Las excepciones que se deben considerar principalmente son aquellas relacionadas con el funcionamiento del sistema, y que están asociadas a:

- Nodos y comunicaciones del particionamiento físico del sistema. Cabe mencionar que estas excepciones se activarán cuando no haya disponibilidad del gestor de base de datos o cuando existen problemas en la comunicación.
- Rangos o valores no válidos en la entrada de datos, tales como atributos obligatorios, con formatos específicos, entre otros.

Se establecen los requerimientos, normas y estándares originados como consecuencia del desarrollo del sistema o adopción de determinada tecnología.

También se describen los requerimientos de operación (hardware, software, comunicaciones), seguridad y control, especificando los procedimientos necesarios por ejecutar después de que el software salga a producción. Como parte de la ejecución se deben proyectar las capacidades de lo siguiente:

- Almacenamiento: consignar el espacio en disco, espacio en memoria, proyección estadística de crecimiento de información.
- Procesamiento: número y tipo de procesadores, memoria.
- Comunicaciones: líneas, capacidad de elementos de red.

Cabe mencionar que la proyección se deberá efectuar basada en data estadística del proceso manual que se viene trabajando.

Como parte de la arquitectura del diseño, se deberá consignar en el documento de diseño lo siguiente:

- Diseño de la arquitectura del sistema (particionamiento físico y descripción de subsistemas)
- Entorno tecnológico y restricciones técnicas
- Lista de excepciones
- Proyección de capacidades
- Procedimientos de operación del sistema (configuración, seguridad, accesos)
- Diagrama de estructura
- Diagrama de interacción de objetos
- Diagrama de paquetes
- Diagrama de clases
- Diagrama de despliegue

Durante la construcción y producto de la interacción del sistema con la infraestructura de soporte o con entidades externas, pueden surgir nuevos subsistemas que deberán ser considerados como parte del documento de diseño.

1.3.2 Diseño de casos de uso

Esta actividad aplica para el modelo orientado a objetos, se especifica el comportamiento del sistema mediante caso de uso, se definen los objetos, las operaciones de las clases e interfaces. Asimismo, de ser necesario se complementan los escenarios de los casos de uso; en este caso se diseñarán las interfaces correspondientes.

El objetivo es identificar las clases que intervienen en cada caso de uso, las cuales se identifican a partir de las clases del análisis y de aquellas clases adicionales necesarias para el escenario que se está diseñando.

Conforme se va elaborando la descripción de los casos de uso, pueden surgir nuevas clases de diseño que no hayan sido identificadas anteriormente y que se incorporan al modelo de clases.

Adicionalmente, se considera el diseño del comportamiento de interface de usuario y formatos de impresión a partir de la especificación de las mismas. En la fase de Análisis, se revisa la interface de usuario, la navegación entre ventanas, los elementos de cada interface, sus características y cómo se gestionan los eventos relacionados con los objetos.

En aquellos casos en los que se realizan modificaciones significativas sobre la interface de usuario, es conveniente que el usuario las valide.

Se deberá generar como parte de esta actividad lo siguiente:

- Diagrama de interacción de objetos
- Diagrama de transición de estados
- Prototipos

1.3.3 Diseño de clases

Esta actividad se realiza para el modelo orientado a objetos. Se transforma el modelo de clases lógico, que proviene del análisis, en un modelo de clases de diseño, especificando los atributos, operaciones, métodos y relaciones entre ellos, consignando los métodos de agregación, asociación o jerarquía.

En esta etapa se busca identificar clases adicionales generales que completen el modelo de clases analizado previamente, considerando lo siguiente:

- El conjunto de clases del análisis puede modificarse con base en las tecnologías utilizadas.
- Cada clase de interface identificada en el análisis se corresponde en el diseño con una clase que proporcione esa interface.
- Las clases de control involucran la coordinación y secuencia entre objetos; en ocasiones contienen lógica de negocio.
- El diseño de las clases del tipo entidad varía según el software utilizado para su elaboración.
- Las clases pueden ser desarrolladas por el programador, adquiridas en forma de bibliotecas, tomadas del entorno de trabajo.

En las clases identificadas se encuentran clases abstractas, que reúnen características comunes a varias clases, se evalúa la jerarquía de clases de modo que cada subclase aumentará su estructura y comportamiento con la clase abstracta de la que hereda.

Posteriormente, se diseñan las asociaciones y agregaciones entre las clases del modelo de clases, revisando la secuencia de mensajes entre objetos. Para definir las asociaciones, se debe considerar lo siguiente:

- Las características de la asociación se detallan según el entorno de desarrollo utilizado.
- Las relaciones bidireccionales se transforman en unidireccionales, para simplificar la implementación del sistema.
- Se modelan las rutas de acceso óptimas entre las asociaciones, con la finalidad de evitar problemas de rendimiento.
- Se analiza la posibilidad de diseñar como clases algunas de las asociaciones.

1.3.3.1 Identificación de atributos de las clases

Otro de los objetivos del diseño de las clases es identificar para cada clase, los atributos, las operaciones que cubren las responsabilidades que se identificaron en el análisis, y la especificación de los métodos que implementan esas operaciones, analizando los escenarios del *diseño de casos de uso*. Una vez que se ha elaborado el modelo de clases, se define la estructura física de los datos correspondientes a ese modelo. Además, en los casos en que sea necesaria una migración de datos de otros sistemas o una carga inicial de información, se realizará su especificación a partir del modelo de clases y las estructuras de datos de los sistemas de origen.

Se deben definir los atributos de las clases; para ello se revisa el modelo de clases, y se verifica si es necesario incluir atributos adicionales. Para cada atributo se define su tipo con formatos específicos, y si existieran también las restricciones asociadas al atributo.

Asimismo, se analiza la posibilidad de convertir un atributo en clase en los siguientes casos:

- El atributo se defina en varias clases de diseño.
- La complejidad del atributo aumente la dificultad para comprender la clase a la que pertenece.

1.3.3.2 Identificación de las operaciones de las clases

Se definen detalladamente las operaciones de cada clase de diseño tomando como base el modelo de clases generado en el análisis, el diseño de los casos de uso y los requerimientos de diseño que pueden aparecer al definir el entorno de desarrollo.

Las operaciones de las clases de diseño surgen para dar respuesta a las responsabilidades de las clases. Asimismo, para definir las interfaces que ofrece esa clase.

Se debe describir cada operación especificando su nombre, parámetros y visibilidad (pública, privada, protegida).

Para aquellos objetos que presenten distintos estados, se deben identificar las operaciones con un diagrama de transición de estados, mostrando cada acción o actividad.

Los métodos pueden especificarse mediante un algoritmo, usando pseudocódigo basado en la secuencia de interacciones del diagrama de interacción en los que la clase aparezca o en la secuencia de transiciones del diagrama de transición de estados.

En algunos casos, esta actividad se realiza directamente en el proceso de construcción, lo cual puede ser apropiado si es que el diseñador procederá a efectuar la construcción. Sin embargo, si son personas o equipos diferentes, detallar el pseudocódigo en el diseño ayudará mucho en la etapa de construcción y minimizará el nivel de iteraciones.

Se deberá generar como parte de la actividad Diseño de clases lo siguiente:

- Diagrama de clases
- Diagrama de transición de estados

1.3.4 Diseño de módulos del sistema

Esta actividad se realiza para el modelo de *diseño estructurado*, se definen los módulos del sistema y la manera en que van a interactuar unos con otros, intentando que cada módulo trate total o parcialmente un proceso.

Se efectúa una descomposición modular de los subsistemas, se diseñan los módulos de consulta, que en ocasiones no se describen en el modelo de procesos, pero sí se mencionan en la lista de requerimientos.

Se analiza el alcance y características de cada proceso a fin de determinar el acceso a la información de bases de datos, implementación de reglas del negocio; se define la presentación de información en los dispositivos de interface con los que el usuario va a interactuar. Este análisis permite identificar los procesos propios del sistema y los servicios comunes.

Posteriormente, se detalla la lógica de negocio del sistema, mediante pseudocódigo; se definen las interfaces entre los módulos de cada subsistema, entre subsistemas y con el resto de los sistemas de información, incluyendo la comunicación de control como los datos propios del sistema.

Para el diseño de las interfaces se debe considerar lo siguiente:

- Los datos o mensajes y formato de los mismos.
- Los valores o rangos de los datos.
- El origen y destino de los datos.
- La información de control y posibles valores.

Las interfaces entre módulos permiten evaluar las necesidades de comunicación entre los distintos nodos, de modo que influyen en el dimensionamiento del entorno tecnológico.

Se efectúa el diseño detallado de la interface de usuario y formatos de *impresión*, según el entorno tecnológico y considerando los estándares y directrices requeridos por la instalación.

Se realizan las adaptaciones oportunas, teniendo en cuenta los requerimientos de rendimiento, de seguridad, la necesidad de alcanzar los tiempos de respuesta establecidos.

Asimismo, se revisa en detalle la navegación entre ventanas y la data que intercambiarán para establecer la secuencia de presentación más apropiada. Se determinan los datos obligatorios y opcionales, y aquellos que requieren un rango de valores predefinido.

Se realiza el diseño de los mensajes de error, mensajes de aviso o advertencia que genera el sistema en función del tipo de acción que realizará el usuario. Asimismo, se diseñan las interfaces de ayuda que proporcionará el sistema en la interacción con el usuario.

Se recomienda que los prototipos sean validados por el usuario.

Se deberá generar como parte de esta actividad lo siguiente:

- Diagrama de estructura
- Prototipos

1.3.5 Diseño físico

Se desarrolla el modelo físico de datos a partir del modelo lógico de datos normalizado o del modelo de clases, en el caso del modelo orientado a objetos.

Previo a la elaboración del modelo físico se evalúa el gestor de base de datos por considerar, la proyección de crecimiento de las entidades, las necesidades de migración y el volumen de datos.

Esta información es necesaria para una mejor implementación del modelo lógico de datos / modelo de clases. Asimismo, para efectuar una estimación del almacenamiento a nivel de hardware.

El objetivo de identificar los caminos de acceso a los datos persistentes del sistema, que son utilizados por los principales módulos / clases de acuerdo con el modelo físico de datos, es de optimizar el rendimiento de los gestores de datos y el consumo de recursos. También, disminuir los tiempos de respuesta en el sistema.

Se recomienda realizar esta tarea para aquellos módulos / clases que reúnan, entre otras, alguna de las siguientes características:

- Datos críticos
- Concurrencia
- Accesos complejos a datos

Para cada módulo / clase se identifican las tablas o ficheros y el tipo de acceso, y el flujo que se seguirá para la obtención de los datos.

Adicionalmente, se efectúa una estimación del número de accesos que se van a realizar considerando la frecuencia y la prioridad del acceso.

Esta información servirá para conocer los accesos excesivamente costosos y redundantes que pueden comprometer el rendimiento final del sistema; por lo tanto, se deberá optimizar el modelo físico de datos, considerando la creación de nuevos accesos, nuevas normalizaciones o particiones del modelo físico de datos.

Luego de optimizado el modelo físico de datos, se debe enfocar en detectar las posibles mejoras en los niveles de rendimiento establecidos y mayor eficiencia del sistema. Como resultado de ello, se podrá efectuar una desnormalización controlada que tenga como finalidad reducir o simplificar el número de accesos a los sistemas de almacenamiento de datos. Ello puede generar lo siguiente:

- Adicionar elementos redundantes (campos de base de datos)
- Definir nuevos caminos de acceso
- Modificar relaciones
- Dividir o unir tablas

En la revisión de la estructura física de datos se debe tener en cuenta lo siguiente:

- Módulos / clases identificados como críticos.
- Estimación de volúmenes de datos y crecimiento por periodo.
- Frecuencia y tipo de acceso.
- Requerimientos relativos al rendimiento, seguridad, confidencialidad y disponibilidad.

Es importante que la desnormalización se lleve a cabo de una forma controlada, para evitar anomalías en el tratamiento de los datos.

Se deberá generar como parte de esta actividad lo siguiente:

- Modelo físico de datos

1.3.6 Generación de especificaciones de construcción

1.3.6.1 Entorno de construcción

Se debe definir el entorno necesario para la construcción. Asimismo, los repositorios de código generado y acceso a los mismos por los miembros del equipo. Se deberá considerar lo siguiente:

- Entorno tecnológico a nivel de servidores, que incluye hardware, software y niveles de comunicación.
- Herramientas de construcción, generadores de código, compiladores, entre otros.
- Estándares de documentación de código.
- Restricciones técnicas del entorno.
- Requerimientos de operación y seguridad del entorno de construcción.
- Lista de servicios y accesos a los mismos.

1.3.6.2 Definición de componentes y subsistemas de construcción

Cada módulo o clase y cada formato individual de interface tienen correspondencia con un componente del sistema; sin embargo, se pueden agrupar o redistribuir módulos o clases en componentes, siguiendo los siguientes criterios:

- Optimización de recursos.
- Características comunes de funcionalidad o de acceso a datos.
- Requisitos de ejecución.
- Metodología de desarrollo; se considera desarrollos por módulos, por funcionalidad, entre otros.

Los subsistemas de construcción y las dependencias entre ellos consideran aspectos relativos a la plataforma de construcción y ejecución. Entre estos aspectos se pueden mencionar:

- Secuencia de compilación entre componentes
- Agrupación de elementos en librerías o *packages* (por ejemplo, DLL en el entorno Windows, *packages* en Java)

Finalmente, se deberá realizar una especificación de cada componente, en pseudocódigo. Igualmente, determinar los elementos o parámetros complementarios a la definición de componentes en función del entorno tecnológico.

Se deberá generar como parte de esta actividad lo siguiente:

- Diagrama de estructura
- Diagrama de componentes
- Diagrama de despliegue

1.3.7 Migración de datos

Se deberá definir el entorno tecnológico (herramientas, software y hardware) en donde se ejecutará la migración de datos y la carga inicial de acuerdo con las necesidades y requerimientos definidos. Se deberá realizar una estimación del volumen de datos y la proyección de crecimiento para definir la infraestructura necesaria para el almacenamiento de datos.

Se deberá definir el procedimiento de migración de los datos considerando las configuraciones necesarias antes de la ejecución de la misma y la verificación de información posterior a la migración. A continuación, se listan algunos procedimientos para tomar en consideración:

- Procedimientos de seguridad
  - Control de acceso a los datos.
  - Copias de seguridad de los procesos.
  - Recuperación de los datos.
  - Contingencias durante el proceso de migración.
- Procedimientos de carga de datos
  - Depuraciones previas de data innecesaria o no válida
  - Procesos de validación
  - Procesos de importación
  - Procesos de carga y prioridades
- Procedimientos de verificación posterior a la migración
  - Verificación de la integridad de la data al culminar la migración.
  - Verificación de la cantidad de data migrada en base de datos.
  - Verificación de la migración en las tablas destino.

Se deberá generar como parte de esta actividad lo siguiente:

- Plan de migración de datos

1.4 Manuales de usuario

En este ítem se define la información que el usuario requerirá para operar el sistema; la misma podría incluir manual funcional de usuario, manual de cálculo, manual de instalación, entre otros.

1.5 Evidencias de pruebas unitarias

Las pruebas unitarias son ejecutadas por el mismo programador; en ese sentido, deberá probar el código que va desarrollando. Las evidencias que consignará como parte del ciclo de vida deben ser parciales pues el objetivo es culminar el desarrollo del sistema, mas no documentar las pruebas. Las pruebas podrán ser manuales o automatizadas. En el mercado existen varios software de pruebas unitarias relacionadas con el código que agilizará mucho más este proceso.

1.6 Evidencias de pruebas integrales

Las pruebas integrales consignan la integración del sistema en su conjunto, es decir, que si el mismo contempla varios módulos, componentes o servicios. Asimismo, interacción con entidades externas, todo debe ser probado en ambiente de desarrollo en su conjunto, tal cual será probado

posteriormente en el ambiente de calidad. Cabe mencionar que al igual que las pruebas unitarias, el objetivo no es documentar estas pruebas sino probar el desarrollo del software integrado; por tanto, la documentación podrá contemplar un porcentaje de las pruebas totales. Se puede utilizar software del mercado para efectuar las pruebas y optimizar los plazos.

1.7 Documento de pase a producción

En este documento se detallan los requisitos de implantación y despliegue en el ambiente de producción; en algunos casos el documento también contempla las especificaciones de implantación y despliegue para los ambientes de calidad interna. Esta información es necesaria para preparar los entornos, ya sea de pruebas o pase a producción, evaluar las dependencias con otros requerimientos y proyectar la carga y volumen de datos futuros. Si es necesaria tecnología adicional a la que se maneja en la organización o configuraciones especiales, estas se deben contemplar como parte del documento.

A continuación se muestran algunos ejemplos de tablas por considerar en el documento de pase:

- Relación de archivos que serán parte del pase

  En la tabla se muestran los ítems mínimos por considerar en un listado de archivos. Para un despliegue en calidad o en producción, se consigna el nombre del archivo, el tamaño del mismo, si la aplicación se instalará en Intranet o también en Internet, se hace referencia al tipo de archivo (extensión) y la versión.

Tabla 10
Ejemplo de relación de archivos

N°	Nombre de Archivo	Tamaño de Archivo	Instalación/ Publicación	Tipo de Archivo	Versión	Destino (Copiar en)
1	P00462017_1.4gi.cfg	350MB	Internet	cfg	1.4	Directorio de Datos

Nota: Tomado de Wong, 2017.

Para efectos de manejo de repositorios y distribución de fuentes, podría utilizarse la siguiente tabla, en donde por efectos de los repositorios se consignarán número de línea base y revisiones.

Tabla 11
Ejemplo de relación de archivos con repositorios

Nombre de Archivo		Origen (Solo Calidad y Producción)			Destino (copiar en)
		(Copiar desde)	Revisión CFG	Línea Base	
Archivo de configuración	servicio-rrhh.ear.cfg	http://120.50.20.191/repositorios/configuraciones01/	49219	5282	Directorio de datos

Nota: Tomado de Wong, 2017.

- Relación de base de datos: se debe considerar la relación de bases de datos, objetos, usuarios y permisos implicados en el pase; de ser necesaria la creación de un nuevo usuario del sistema, se deben gestionar los permisos necesarios.

Tabla 12
Ejemplo de relación de base de datos (aplicaciones web)

Nombre de la Base de Datos	Computador	Data-Source	Jndi	Instancia App	Dominio	Indicador (Creación/ Eliminación)	Nombre Controlador de la Base de Datos
Recurs	rec02s2	dcrecurs0j	jdbc/dcre-curs0j	Clúster In-tranet-V	Centaurus	C	Oracle 10g (Type 4)

Nota: Tomado de Wong, 2017.

- Relación de contextos de invocación, mediante los cuales se invocará a la aplicación (ruta base de aplicación).

Tabla 13
Ejemplo de relación de contextos

Nombre del Archivo	Descripción de la aplicación	Contexto	Ruta de Invocación
servicio-rrhh.ear	iarrhh.war		
	Acceso Directo: Recursos	/ol-ti-iarrhh/	/mrec007Alias

Nota: Tomado de Wong, 2017.

1.8 Plan de pruebas

Este documento sirve como base para definir las pruebas del sistema, permite efectuar una trazabilidad entre los requisitos y el sistema desarrollado, validando de esta forma las necesidades del usuario.

El plan de pruebas define el objetivo de las pruebas, las estrategias por considerar que involucran un trabajo lineal o paralelo, dependiendo de las necesidades de la organización. Esta actividad puede ejecutarse durante la etapa de análisis o al finalizar la construcción. Si se efectúa en la etapa de análisis puede ser susceptible a cambios posteriores a la construcción; por lo tanto, el riesgo debe ser considerado.

El plan de pruebas también debe contemplar el entorno de pruebas, el mismo que debe ser similar al ambiente de producción, un ambiente en donde se puedan efectuar las pruebas desde el despliegue hasta la ejecución de la operación del software. En el plan de pruebas se deberá definir lo siguiente:

- Entorno de pruebas y hardware necesario
- Configuraciones para el despliegue de la aplicación
- Preparación de data para la ejecución de pruebas
- Estrategia de pruebas
- Tipos de pruebas por ejecutar
- Criterios de aceptación del software

1.9 Casos de pruebas

Este documento incluye todas las pruebas por ejecutarse posteriores a la construcción. Se deberán definir los datos de entrada y salida, así como las operaciones. Los casos de prueba deben tener correspondencia con el documento de análisis y diseño del sistema, es decir, debe haber trazabilidad entre los requisitos del usuario y el software generado. Se debe garantizar que todos los requisitos sean satisfechos.

Se definen también los tipos de pruebas por ejecutar, las cuales pueden ser de regresión, disponibilidad, de seguridad, entre otras.

1.10 Evidencias de pruebas

Las evidencias de pruebas consignan las pantallas de ejecución de los casos de prueba. En este documento se indicarán aquellos casos de prueba ejecutados satisfactoriamente y sus resultados. Asimismo, aquellos que evidenciaron error en su ejecución y que deberán ser revisados por el equipo de desarrollo.

1.11 Informe de Calidad

En esta actividad se analizan y evalúan los resultados de las pruebas con la finalidad de conocer el grado de cumplimiento, número de fallas, número de iteraciones, entre otros. Para ello se efectúa lo siguiente:

- Comparación de los resultados obtenidos con los esperados, según el umbral de la métrica establecida.
- Identificación de la causa de cada problema detectado para tomar las acciones correctivas necesarias.

Se deberá definir si el plan de pruebas y los casos de pruebas deben volver a realizarse total o parcialmente, y si será necesario incluir nuevos casos de pruebas.

Lectura seleccionada n.° 5

Gómez Fuentes, M. (2011). *Notas del curso: Análisis de Requerimientos*. México D.F.: Universidad Autónoma Metropolitana. Recuperado de http://bit.ly/2c7KsJc

Actividad n.° 5

Foro de discusión sobre el análisis de requerimientos de software.

Instrucciones

- Ingrese al foro y participe con comentarios críticos y analíticos sobre el tema análisis de requerimientos.
- Lea y analice el tema N° 1 del manual.
- Responda en el foro a las preguntas acerca del análisis de requerimientos de software: Mencione tres metodologías de desarrollo, ventajas y desventajas.

Técnicas de especificación de requerimientos

Tema n.° 2

Las técnicas de especificación de requerimientos se utilizan para describir los requerimientos especificados por el usuario. Es necesario cubrir todos los requisitos funcionales, o funcionales y técnicos, a fin de implementar el producto de software requerido.

1. Especificación de requerimientos de software – IEEE 830

IEEE Std 830 (1998) menciona algunos aspectos por considerar como parte de la especificación de requerimientos de software, incluyendo lo siguiente (p. 003):

- Naturaleza de la especificación de requerimientos de software
- Medio ambiente de la especificación de requerimientos de software
- Características de la especificación de requerimientos de software
- Preparación conjunta de la especificación de requerimientos de software
- Evolución de la especificación de requerimientos de software
- Prototipado
- Diseño de inclusión en la especificación de requerimientos de software
- Incorporación de los requisitos del proyecto de especificación de requerimientos de software

1.1. Naturaleza de la especificación de requerimientos de software

Refiere a especificaciones del producto de software. Los aspectos básicos que deben considerarse son funcionalidad, interfaces externas, rendimiento, atributos, y restricciones de diseño.

1.2. Medio ambiente de la especificación de requerimientos de software

El IEEE Std 830 (1998) menciona:

El software puede contener esencialmente toda la funcionalidad del proyecto o puede ser parte de una Sistema más grande. En este último caso típicamente habrá una especificación de requerimiento de software que indicará las interfaces entre el sistema y su porción de software, y colocará requisitos de rendimiento y funcionalidad externos (p. 004).

Adicionalmente, se deberán considerar dependencias que puedan existir a nivel de aplicación, base de datos o servicios. La especificación de requisitos contempla requisitos a nivel funcional y no técnicos; el detalle técnico se deberá contemplar en las especificaciones de diseño del software.

1.3 Características de la especificación de requerimientos de software

La especificación de requerimientos del software deber ser correcta, no debe poseer ambigüedad, debe ser completa, consistente, clasificada por importancia y/o estabilidad, verificable, modificable, trazable (IEEE Std 830, 1998, p. 004).

1.4 Preparación conjunta de la especificación de requerimientos de software

El proceso de desarrollo de software debe comenzar con el acuerdo entre el proveedor y el cliente sobre lo que el Software completado debe hacer. Este acuerdo, en forma de especificación de requerimiento de software, debe ser preparado conjuntamente. Esto es importante porque ge-

neralmente ni el cliente ni el proveedor están calificados para escribir una buena especificación solos (IEEE Std 830, 1998, p. 008).

Es importante la elaboración de actas de trabajo que describan los acuerdos cliente-proveedor, así como contemplar la aprobación del usuario al finalizar el detalle de la especificación de requerimientos en el documento de análisis.

1.5 Evolución de la especificación de requerimientos de software

Las especificaciones de los requerimientos evolucionan a medida que se va avanzando con las especificaciones del análisis, razón por la cual es necesario que el usuario participe y revise cada especificación antes de que pase a la fase de diseño.

1.6 Prototipado

IEEE Std 830 (1998), precisa que los prototipos son útiles por las siguientes razones:

- Es más probable que el cliente vea el prototipo y reaccione ante él que al leer la especificación de requerimientos de software. Por lo tanto, el prototipo proporciona retroalimentación rápida.
- El prototipo muestra aspectos imprevistos del comportamiento de los sistemas. Así, produce no solo respuestas sino también nuevas preguntas. Esto ayuda a cerrar la especificación de requerimientos de software.
- Una especificación de requerimientos de software basada en un prototipo tiende a sufrir menos cambios durante el desarrollo, acortando así el tiempo de desarrollo (p. 9).

1.7 Diseño de inclusión en la especificación de requerimientos de software

Un requisito especifica una función o atributo visible externamente de un sistema. Un diseño describe un subcomponente particular de un sistema y / o sus interfaces con otros subcomponentes. El (los) escritor (es) SRS debe (n) distinguir claramente entre identificar las restricciones de diseño requeridas y proyectar un diseño específico (IEEE Std 830, 1998, p. 9).

1.8 Incorporación de los requisitos del proyecto de especificación de requerimientos de software

Las especificaciones de requerimientos de software se deben enfocar en contemplar todos los elementos necesarios para el desarrollo del producto de software. Los temas relacionados con el tema contractual o de gestión de proyecto no deben ser parte de estas especificaciones.

Lectura seleccionada n.° 6

Rational Software. (1998). *Rational Unified Process. Best Practices for Software Development Teams*. Recuperado de https://ibm.co/1k6R9DE

Actividad n.° 6

Foro de discusión sobre análisis de requerimientos de software.

Instrucciones:

- Ingrese al foro y participe con comentarios críticos y analíticos del tema análisis de requerimientos.
- Lea y analice el tema N° 2 del manual.
- Responda en el foro a las preguntas acerca del análisis de requerimientos de software: Mencione tres herramientas utilizadas en el análisis de requerimientos, y sus beneficios.

Tarea académica n.° 3

I. Trabajo de investigación

a. Seleccione un proyecto de automatización de software en plataforma web y desarrolle lo siguiente:
   i. Modelamiento de la situación actual
   ii. Modelamiento de la situación propuesta
   iii. Diagrama de casos de uso, deberá incluir la descripción de cada CUS.
   iv. Prototipos de la aplicación, que incluyan la descripción de la navegabilidad.

b. Sobre la presentación:
   i. Presentar en archivo Word; se deberán pegar las pantallas generadas y los prototipos.
   ii. Puede usar imágenes y escribir palabras que considere importantes.

II. Rúbrica

n.°	Rúbrica	Min. puntaje	Máx. puntaje
1	Modelamiento de la situación actual	0	3
2	Modelamiento de la situación propuesta	0	3
3	Diagrama de casos de uso	0	8
4	Prototipos	0	6

III. Indicaciones

- Todos los puntos de la rúbrica deberán ser subidos a la plataforma en formato ZIP y como nombre del archivo deberán tener el siguiente formato: **APELLIDO_NOMBRE.zip**
- En el caso de plagio, el trabajo será invalidado y la nota será de 00.

Glosario de la Unidad III

A

Arquitectura de software: Modelo de la estructura y organización fundamental de un sistema de software (Sommerville, 2011).

C

Caso de uso: Especificación de un tipo de interacción con un sistema (Sommerville, 2011).

Calidad: El grado en que un conjunto de características inherentes cumple con los requisitos (IIBA, 2009).

Ciclo de vida del software: Utilizado a menudo como otro nombre para el proceso del software. Originalmente acuñado para referirse al modelo en cascada del proceso del software (Sommerville, 2011).

Clase objetos: Una clase de objetos define los atributos y operaciones de los objetos. Los objetos se crean en tiempo de ejecución mediante la instanciación de la definición de la clase. El nombre de la clase de objetos se puede utilizar como un nombre de tipo en algunos lenguajes orientado a objetos (Sommerville, 2011).

S

Sistema: Una colección de subsistemas que están interrelacionados e interdependientes, trabajando juntos para lograr metas y objetivos predeterminados. Todos los sistemas tienen entrada, procesos, salida y retroalimentación (Sommerville, 2011).

T

Tolerancia a defectos: Capacidad de un sistema para continuar en ejecución incluso después de que hayan tenido lugar a defectos (Sommerville, 2011).

U

Unidad de software: Pieza de código compilado por separado (Indecopi, 2006).

V

Verificación: Proceso de verificar que un sistema cumple su especificación (Sommerville, 2011).

Bibliografía de la Unidad III

Arisholm, E., Gallis, H., Dyba, T., & Dag I.K. Sjoberg. (2007). Evaluating pair programming with respect to system complexity and programmer expertise. *IEEE Transactions on Software Engineering, 33*(2), 65-86. Disponible en http://ieeexplore.ieee.org/document/4052584/

Bourque, P., & Fairley, R. (2014). *SWEBOK V3.0. Guide to the Software Engineering Body of Knowledge*. New Jersey: IEEE. Disponible en http://bit.ly/2zcSdFX

Gómez Fuentes, M. (2011). *Notas del curso: Análisis de Requerimientos*. México D.F.: Universidad Autónoma Metropolitana. Recuperado de http://bit.ly/2c7KsJc

Institute of Electrical and Electronics Engineers. (1998). *IEEE Recommended Practice for Software Requirements Specifications*. Nueva York: autor. Recuperado de http://bit.ly/2bbzlHv

Kendall, K., & Kendall, J. (2013). *Systems analysis and design* (9a ed.). New Yersey: Prentice Hall.

Pressman, R. (2005). *Ingeniería del Software. Un enfoque práctico* (6a ed.). D.F., México:McGraw-Hill.

Rational Software. (1998). *Rational Unified Process. Best Practices for Software Development Teams*. Recuperado de https://ibm.co/1k6R9DE

Sommerville, Ian. (2011). *Software Engineering* (9a ed.). Madrid: Pearson Education.

Wiegers, K., & Beatty, J. (2013). *Software Requirements* (3a ed.). Washington: Microsoft Press. Disponible en http://bit.ly/2ghNk6M

Autoevaluación n.º 3

Marca la alternativa correcta:

1. ¿Cuál de las siguientes No es una metodología de desarrollo de software?
   a) JAVA
   b) Xtreme programming
   c) SCRUM
   d) Microsoft Solution Framework
   e) Análisis y diseño de sistemas estructurados

2. Cuál de las siguientes No es una fase de la Metodología RUP.
   a) Transición
   b) Construcción
   c) Inicio
   d) Potenciación
   e) Elaboración

3. La metodología RUP es más adaptable a proyectos de:
   a) Mediano plazo
   b) Corto plazo
   c) Largo plazo
   d) Mediano y corto plazo
   e) Muy corto plazo

4. ¿Es SCRUM una metodología ágil para el desarrollo de software?
   a) No
   b) Depende del tamaño del proyecto
   c) Depende del tiempo de duración del proyecto
   d) Sí
   e) Depende de los riesgos del proyecto

5. ¿Cuál es el mejor modelo a seguir cuando se ha implementado muchas veces el mismo producto de software?
   a) DRA
   b) Cascada
   c) Incremental
   d) Espiral
   e) SCRUM

6. La conversión de los datos en una forma más significativa se llama:
   a) Procesamiento
   b) Captura
   c) Carga
   d) Retroalimentación
   e) Organizador

7. Es un sistema de información que utiliza la tecnología de visualización de datos para analizar y visualizar los datos para la planificación y la toma de decisiones de mapas digitalizados.
   a) DSS
   b) MIS
   c) GIS
   d) TPS
   e) SIG

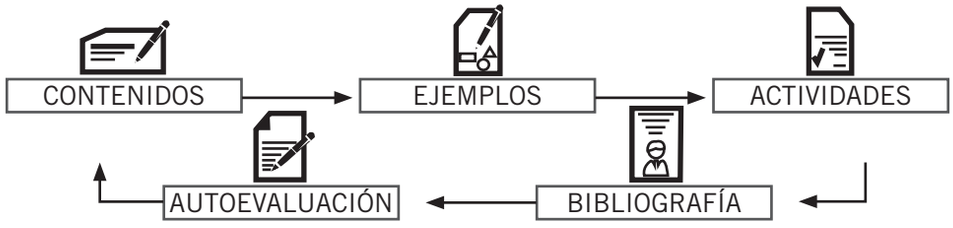
8. Las entrevistas y reuniones de trabajo son:
   a) Una herramienta de análisis
   b) Una técnica para la obtención de requerimientos
   c) Una metodología de requerimientos
   d) Una herramienta de gestión de proyecto
   e) Una técnica de requerimiento de interface

9. Son las actividades sugeridas para un proceso de requerimientos:
   a) Obtención de requerimientos
   b) Negociación y análisis de requerimientos
   c) Validación de requerimientos
   d) Verificación de requerimientos
   e) Todas las anteriores

10. Es una de las características que hace único al RUP:
    a) Desarrollo iterativo e incremental
    b) Funcionalidad
    c) Rapidez
    d) Fácil de usar
    e) Es conocido

UNIDAD IV
VALIDACIÓN DE REQUERIMIENTOS

DIAGRAMA DE PRESENTACIÓN DE LA UNIDAD IV



ORGANIZACIÓN DE LOS APRENDIZAJES

RESULTADO DE APRENDIZAJE: Al finalizar la unidad, el estudiante será capaz de validar la especificación de requerimientos de software de su proyecto.

CONOCIMIENTOS	HABILIDADES	ACTITUDES
Tema n.° 1: Revisiones e inspecciones 1. Revisiones 2. Inspecciones **Tema n.° 2: Prototipos para validar requerimientos** 1. Validación de prototipos 1.1. Ventajas del prototipo 1.2. Desventajas del prototipo **Tema n.° 3: Prueba de aceptación de diseño** 1. Validación de la definición de la arquitectura 2. Validación de la arquitectura de datos 3. Validación de las especificaciones de construcción **Lectura seleccionada n.° 7** "3.2 Review Process (K2)" (Müller & Friedenberg, 2011, pp. 33-35). **Tema n.° 4: Validación de los atributos de calidad del producto** 1. Pruebas unitarias 2. Pruebas integrales 3. Pruebas de sistemas 3.1. Pruebas de caja blanca 3.2. Pruebas de caja negra 3.3. Pruebas de regresión 3.4. Pruebas no funcionales 4. Pruebas de aceptación 5. Pruebas automatizadas 5.1. Selenium 5.2. JMeter 5.3. TestLink 5.4. PMD 5.5. Check Style 5.6. SONARQUBE 5.7. Bugzilla 5.8. Appium 5.9. Mantis 5.10.Jenkins 6. Ciclo de vida de desarrollo con el uso de herramientas	1. Detectar anomalías en especificación de requerimientos de software. **Actividad n.° 7** Los estudiantes participan en el foro de discusión sobre proceso de revisión. **Actividad n.° 2** Los estudiantes participan en el foro de discusión sobre técnicas de pruebas. **Tarea académica n.° 4**	1. Valora el documento de la especificación de requerimientos de software.

Tema n.° 5: Análisis de la interacción de requerimientos

1. Definición de requerimiento
2. Análisis de factibilidad de un requerimiento
3. Interacción de requerimientos

Tema n.° 6: Análisis formal de requerimientos

1. Características de la especificación de requisitos de software
2. Análisis formal

Lectura seleccionada n.° 8

Certified Tester Foundation Level Syllabus. Version 2011 (Müller & Friedenberg, 2011).

Autoevaluación de la Unidad IV

Revisiones e inspecciones

Tema n.° 1

En el siguiente capítulo, conocerá los diferentes conceptos en relación con las revisiones e inspecciones, las cuales constituyen en su conjunto verificaciones de calidad de los productos generados como parte del ciclo de vida del desarrollo del software.

1. Revisiones

La IEEE menciona que el propósito de las revisiones es:

> Evaluar un producto de software por un equipo de personal calificado para determinar su idoneidad para el uso previsto e identificar discrepancias con las especificaciones y estándares (IEEE Standard for Software Reviews, 1998, p. 9).

Por tanto, se puede determinar que la revisión se da en contexto más general tanto en el control de calidad como en el aseguramiento de calidad. Asimismo, se pueden definir revisiones formales e informales orientadas a la verificación del producto y del proceso.

En una revisión técnica, el propósito principal es identificar discrepancias entre el producto y las especificaciones base. Los productos que pueden estar sujetos a una revisión técnica son aquellos que constituyen parte del ciclo de vida; por ejemplo:

- Modelo de negocio
- Análisis funcional y técnico
- Diseño
- Documentación de pruebas
- Manuales de usuario

2. Inspecciones

La *Guía de los fundamentos para la dirección de proyectos* (Guía del PMBOK) consigna a la *inspección* como una técnica de validación y precisa lo siguiente:

> La inspección incluye actividades tales como medir, examinar y validar para determinar si el trabajo y los entregables cumplen con los requisitos y los criterios de aceptación del producto. Las inspecciones se denominan también, revisiones, revisiones del producto, auditorías y revisiones generales. En algunas áreas de aplicación, estos diferentes términos tienen significados singulares y específicos (Project Management Institute, 2013, p. 134).

La inspección es un tipo de revisión y es más específico pues se relaciona directamente con el producto o entregable o con una actividad misma.

La inspección es una técnica de revisión formal que debe ser dirigida por un moderador especialista en la materia por inspeccionar. La organización del equipo de inspección debe estar definida con roles específicos con el fin de cumplir el procedimiento desarrollado para llevar a cabo la inspección, y documentar la inspección mediante listas de verificación, métricas, listas de hallazgos, informe de inspección, entre otros.

El propósito principal de la inspección es encontrar defectos. Si es llevada a cabo internamente en la organización por roles del mismo nivel organizacional, a esta revisión se le denominará "revisión por pares".

Prototipo para validar requerimientos

Tema n.° 2

La elaboración de prototipos es una técnica de validación de requisitos que involucra el diseño de pantallas en alguna herramienta gráfica o el maquetado de una aplicación en HTML, de modo que el usuario final pueda validar las interfaces del software requerido antes de la construcción y así prever cómo quedará el software. En algunos casos, el maquetado involucra navegabilidad de la aplicación; eso dependerá del tipo de metodología y aplicación que se va a desarrollar. Generalmente, esto se puede observar en el desarrollo de aplicaciones móviles.

1. Validación de prototipos

La presentación de interfaces gráficas se presenta en los documentos de análisis funcional, y como buena práctica se convoca al usuario o usuarios finales a fin de que puedan otorgar una validación y de ser necesario una retroalimentación para mejorar los prototipos. Resulta importante que los prototipos sean validados en las etapas tempranas del ciclo de vida del desarrollo del software, pues minimiza los riesgos de modificación luego de la etapa de construcción.

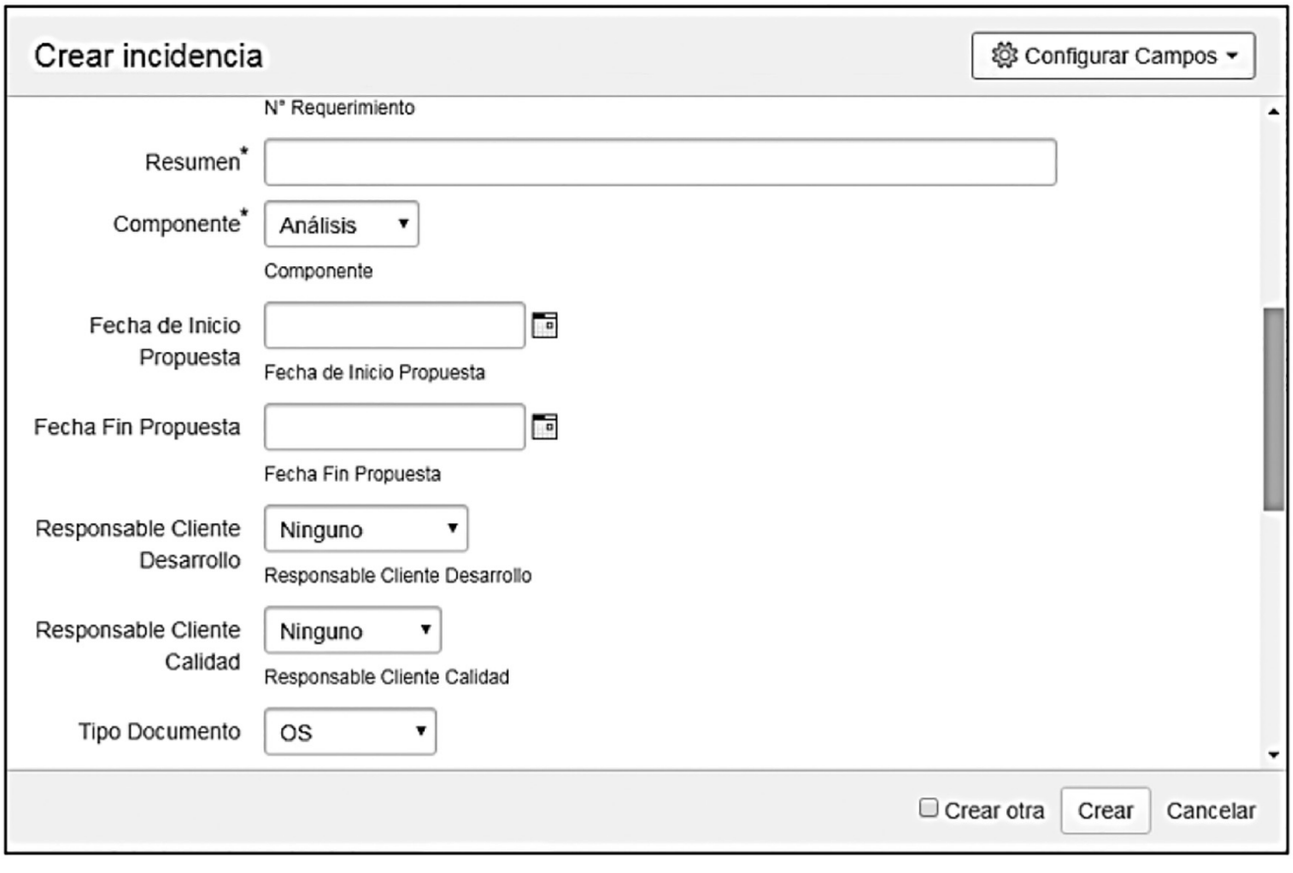


Figura 39: Ejemplo de prototipo. Fuente: Wong, 2017.

1.1 Ventajas del prototipo

- Minimiza los riesgos de generación de software que no satisfaga la necesidad del usuario.
- Reduce el costo de iteración e incrementa la probabilidad de éxito del proyecto.
- Prevé el uso de herramientas para el desarrollo de software.

- Muestra al usuario final una vista rápida del desarrollo requerido.
- Permite identificar funcionalidades no contempladas, para su inclusión en el análisis.
- Apoya en la navegabilidad de la aplicación.
- Provee un mecanismo de consenso de las interfaces de la aplicación.

1.2 Desventajas del prototipo

- El proceso de elaboración de prototipos puede tomar mucho tiempo si el usuario final no tiene claro lo que desea, y no toma una decisión rápida en la validación.
- Cuando el prototipo es desarrollado en HTML o móvil, el usuario final piensa que la aplicación está terminada y no otorga mucho tiempo para el desarrollo de la misma.
- Durante la etapa de desarrollo, el programador puede alterar el prototipo y generar una iteración que requiera nuevamente la validación del usuario y, en consecuencia, un incremento en costo y tiempo.

Prueba de aceptación de diseño

Tema n.° 3

El diseño presenta en un nivel más técnico el desarrollo del sistema definiendo su arquitectura y el detalle de los componentes de software.

Las pruebas de aceptación del diseño involucran la revisión los puntos que mencionamos a continuación.

1. Validación de la definición de la arquitectura

- Mediante el diagrama de despliegue se valida el diseño de la arquitectura.
- Se efectúa una trazabilidad de las especificaciones de diseño versus los requisitos funcionales y requisitos no funcionales; de esta manera se valida la consideración de todos los puntos que van a ser producto de evaluación en el software.
- Se precisan las excepciones del alcance, que no serán consideradas como parte del desarrollo del software. Estas pueden ser excepciones de base de datos, de aplicación, de comunicación, configuraciones en el servidor, capa de presentación del software.
- Se validan los elementos de infraestructura considerando el hardware, software y comunicaciones, definiendo de esta manera el entorno tecnológico.
- Se valida la proyección de almacenamiento de información por base de datos. Se debe considerar como base la información estadística recopilada en la etapa de análisis.
- Se validan las transacciones a nivel de aplicación que se consignarán en la aplicación.
- Se validan los servicios con los que va a interactuar la aplicación, o aquellos que se van a desarrollar como parte del requerimiento.
- Validación de la arquitectura del software de acuerdo con el tipo de desarrollo:
    - Estructurado, se validará el diagrama de estructura.
    - Orientado a objetos, se validará el diagrama de clases, diagrama de casos de uso, diagrama de secuencia o diagrama de colaboración.

1.1 Validación de la arquitectura de datos

- Se valida el modelo físico, se consideran los nombres de la tablas y atributos contemplados como parte del desarrollo de software.
- Se validan las operaciones de base de datos por ejecutarse y la secuencia de las mismas.
- Se validan las transacciones que se consignarán en la aplicación. Debe haber una correspondencia con el estándar de la organización.

1.2 Validación de las especificaciones de construcción

- Se validan las especificaciones de entorno de la aplicación; considera los servidores en donde se desarrollará la aplicación, el sistema operativo, lenguaje de programación, base de datos y configuraciones requeridas para el despliegue de la aplicación.

- Se validan las especificaciones de construcción por cada componente por desarrollar. La especificación en algunos casos puede ser hasta pseudocódigo.

- Se validan los parámetros de entrada y salida de servicios por considerar como parte del desarrollo de la aplicación.

Las pruebas de aceptación del diseño es documental, son verificadas por un equipo de control de calidad interno del proyecto y validadas por personal técnico externo. Parte de las técnicas utilizadas son las revisiones de pares que implican el uso de *checklist*, la detección de no conformidades y gestión de las mismas hasta que sean levantadas.

Lectura seleccionada n.º 7

Leer apartado "3.2 Review Process (K2)", pp. 33-35.

Müller, T., & Friedenberg, D. (2011). *Certified Tester Foundation Level Syllabus. Version 2011*. Disponible en http://bit.ly/2fvGrib

Actividad n.º 7

Foro de discusión sobre proceso de revisión.

Instrucciones:

- Ingrese al foro y participe con comentarios críticos y analíticos del tema Proceso de revisión.
- Lea y analice los temas 1, 2 y 3 del manual.
- Responda en el foro a las preguntas acerca del modelado de sistemas:
  Mencione tres casos en donde se pueden aplicar las técnicas de revisión.

Validación de los atributos de calidad del producto

Tema n.° 4

Los atributos de calidad del producto pueden ser verificados y validados mediante diferentes pruebas. A continuación se mencionan diversos tipos de pruebas que se pueden considerar en esta fase.

1. Pruebas unitarias

Las pruebas unitarias son ejecutadas en la etapa de construcción por el programador, involucra verificar el correcto funcionamiento del código desarrollado y que este cumpla con las especificaciones y estándares definidos. Las pruebas unitarias deben contemplar la revisión del total de código desarrollado, pues debe garantizarse que el software funcione correctamente. Asimismo, debe considerarse que de acuerdo con las necesidades del usuario, el software puede tener posteriores mantenimientos de código y, de ser necesario, parte de su código ser reutilizado para otras aplicaciones; un ejemplo de esto es el consumo de servicios.

Para la ejecución de estas pruebas se pueden utilizar las siguientes herramientas: JUNIT, CACTUS, SONAR.

2. Pruebas integrales

Las pruebas integrales son pruebas de validación generadas en la fase de desarrollo, se ejecutan luego de las pruebas unitarias y validan el software en su conjunto, así como la interacción con sistemas satelitales y servicios. Estas pruebas consignan la revisión de los requisitos funcionales y no funcionales de la aplicación.

3. Pruebas de sistemas

Las pruebas de sistemas son pruebas de validación que se ejecutan en la etapa de *pruebas de calidad*, en que se ejecuta el despliegue del software en un ambiente similar al de producción siguiendo las configuraciones especificadas en el documento de pase a producción. Posteriormente se revisa el correcto funcionamiento del software basándose en los casos de pruebas que consignan los requisitos funcionales y no funcionales del requerimiento. A continuación, mencionamos algunas pruebas que se ejecutan en esta etapa.

3.1. Pruebas de caja blanca

Las pruebas de caja blanca son técnicas y validan estándares de programación, código de la aplicación, comportamiento de la base de datos y parámetros de entrada y salida de los servicios. Se sugiere que el personal de pruebas de control de calidad tenga experiencia en desarrollo de software y conocimiento de programación.

La desventaja de este tipo de pruebas es que no pueden identificar falta de funcionalidad por flujos alternativos u omisiones al código.

Para la ejecución de estas pruebas se pueden utilizar las siguientes herramientas:

- PMD
- Check Style
- SONAR
- Simian
- Google CodePro Analytix

3.2. Pruebas de caja negra

Las pruebas de caja negra son netamente funcionales, validan todos los requisitos funcionales con parámetros de entradas predefinidas, verifican el comportamiento de la aplicación y evalúan las salidas. Este tipo de pruebas es complementario a las pruebas de caja blanca, ya que considera un enfoque diferente de revisión.

El objetivo de la prueba de caja negra es identificar las diferencias existentes entre las especificaciones funcionales y el software.

Para la ejecución de estas pruebas se pueden utilizar las siguientes herramientas:

- Quick TestPro
- Rational Robot
- Test Complete
- QA Wizard
- Selenium
- Soapui
- Watir (aplicaciones web programadas en Ruby)
- WatiN (aplicaciones programadas en .Net)
- Solex
- Imprimatur

A continuación, se muestra gráficamente el proceso de la prueba de caja negra, en donde se identifican datos de entradas y se evalúan los datos de salida. En este caso no se evalúa el proceso de operación.

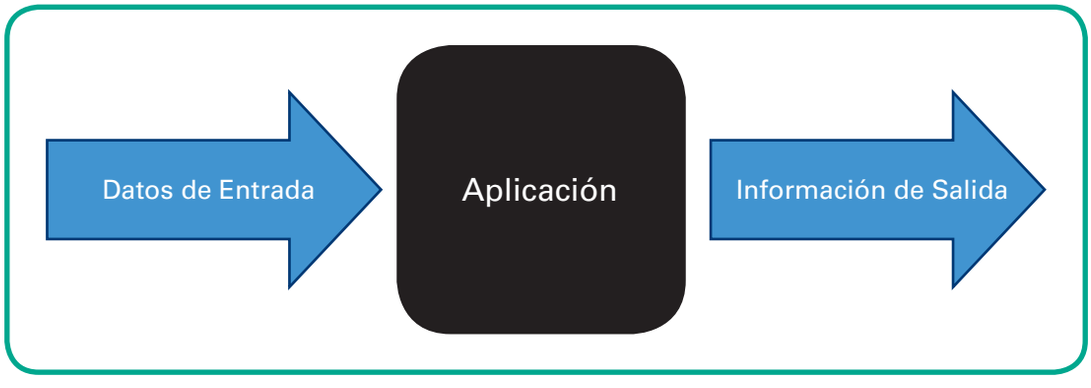


Figura 40: Pruebas de caja negra. Fuente: Wong, 2017.

3.3. Pruebas de regresión

Las pruebas de regresión son mediciones de validación efectuadas sobre aplicaciones que ya han sido ensayadas previamente. Están orientadas a efectuar una prueba funcional completa de la aplicación. En el caso de los mantenimientos de aplicación no solo contemplan las especificaciones funcionales del requerimiento, sino también la revisión de toda la aplicación en su conjunto. El objetivo de esta prueba es identificar defectos después de que se ha efectuado una modificación al software; por tanto, pueden identificarse defectos de la modificación efectuada o de otra funcionalidad del software. El criterio de selección de esta prueba consiste en mitigar riesgos de la ejecución del software posterior a la modificación.

En algunos casos las pruebas de regresión pueden incluir revisiones técnicas. Esto va a depender del nivel de criticidad de la aplicación dentro de la organización.

3.4. Pruebas no funcionales

Las pruebas no funcionales son validaciones de la operación de la aplicación consignando diversos criterios que garanticen la calidad del producto.

Las pruebas no funcionales tienen como objetivo realizar comprobaciones que garanticen la calidad del producto desde el punto de vista técnico y, al mismo tiempo, permitan conocer con anterioridad las capacidades y riesgos relacionados con la performance de la aplicación que será instalada en producción.

Los criterios que se consideran en las pruebas no funcionales están relacionadas con pruebas de estrés, concurrencia, configuración, volumen, carga, usabilidad, escalabilidad, seguridad, entre otras. Cabe mencionar que estas pruebas pueden ser ejecutadas con herramientas de automatización orientadas a forzar la aplicación según ciertas casuísticas. A continuación, mencionaremos los tipos de pruebas *no funcionales* que son utilizados más frecuentemente:

- Pruebas de seguridad: orientadas a validar la vulnerabilidad del sistema y el entorno de configuración a nivel de hardware. Se revisan las características asociadas a la seguridad, tales como cumplimiento de estándares de programación, control de acceso al sistema, canales seguros mediante la red, configuración de servidores y servicios, protección de páginas web que tienen exposición pública, protección de robots que consultan las bases de datos, hosting de almacenamiento de la información.

Entre las herramientas que tenemos para este tipo de pruebas mencionaremos las siguientes:

- Wireshark
- Kali Linux
- NMAP (Network Mapper)
- Anubis

- Pruebas de volumen: estas pruebas validan la capacidad del sistema para el soporte de manejo de gran volumen de datos en base de datos. Se efectúan pruebas de concepto con data de entrada en grandes volúmenes, se mide el tiempo de procesamiento de data, se efectúan consultas y verifican respuestas de la base de datos, se evalúan las restricciones de tiempo y almacenamiento. El objetivo de esta prueba es delimitar el grado de carga de información, tiempos de respuesta y almacenamiento de datos.

- Pruebas de carga: evalúan el comportamiento del sistema frente a una sobrecarga de datos durante un período predeterminado; se repiten diversas acciones de entrada de datos para evaluar la respuesta del sistema. El objetivo de esta prueba es evaluar que el sistema funcionará correctamente en todo momento.

Para la ejecución de estas pruebas se pueden utilizar las siguientes herramientas:

- FunkLoad
- FWPTT Load Testing
- LoadUI
- Jmeter

- Pruebas de usabilidad: en estas pruebas se evalúa la facilidad del uso del sistema, nivel de intuición para poder navegar de una pantalla a otra, qué tan amigable y fácil será para el usuario final poder trabajar con la aplicación. Hoy en día con el manejo de las aplicaciones móviles, internet y redes sociales, estas pruebas han cobrado mucho más relevancia.

- Pruebas de escalabilidad: buscan probar la capacidad de escalamiento de alguna no funcionalidad del sistema; por ejemplo, la carga, volumen de datos, manejo de transacciones, entre otras. En este caso se hacen las pruebas por bloques para evaluar la capacidad de cada característica no funcional.

- Pruebas de estrés: son pruebas de carga que fuerzan al sistema con una sobrecarga operativa con la finalidad de determinar el punto de quiebre. El objetivo de esta prueba es conocer la capacidad del sistema para saber el nivel de disponibilidad que tendrá la aplicación frente a determinados escenarios.

- Pruebas de configuración: estas pruebas son verificadas en diversos entornos de hardware y software; para ello se consideran diferentes versiones de software, navegadores, sistemas operativos, diferentes capacidades de hardware, diferentes dispositivos. El objetivo de esta prueba es que el sistema tenga el mismo comportamiento en todos los entornos configurados. Asimismo, determinar la línea base en software y hardware para su operación. Considerando la evolución de la tecnología y los continuos cambios que se vienen dando en los lenguajes de programación, esta prueba es muy importante y hasta a veces crítica, sobre todo en las aplicaciones web y móviles.

- Pruebas de concurrencia: refieren a la simultaneidad de operaciones en un sistema de manera concurrente, se simula el empleo del sistema por varios usuarios ejecutando diversas operaciones. El objetivo de esta prueba es evaluar el número de operaciones en un intervalo de tiempo, el pico máximo de operaciones, validación de las conexiones físicas que asignan la demanda a los servidores (balanceadores).

Entre las herramientas más utilizadas, tenemos:

- Hp LoadRunner
- IBM Rational Performance Tester
- Borland SilkPerformer
- WebLoadProfessional
- ANTS-Advanced .NET Testing System
- Test Studio
- LoadStorm
- Jmeter

4. Pruebas de aceptación

Las pruebas de aceptación son ejecutadas por el usuario final, evalúan el funcionamiento del software según los requisitos funcionales y no funcionales especificados en el documento de análisis. Como buena práctica, se prepara un documento de aceptación de pruebas de usuario que muestra el ciclo de pruebas que el usuario deberá seguir para otorgar la aceptación; sin embargo, la misma no debe ser restrictiva pues no hay una secuencia específica que siga el usuario al momento de ejecutar las pruebas, ya que la ejecuta de acuerdo con su experiencia, considerando en ocasiones casuísticas diferentes a las especificadas en los documentos.

El objetivo de esta prueba es que el usuario examine el software desde el punto de vista funcional y otorgue su aceptación.

5. Pruebas automatizadas

Las pruebas automatizadas permiten agilizar el flujo de pruebas mediante el uso de herramientas; en las mismas se registran criterios de pruebas y, según esto, las ejecutan. La ventaja del uso de herramientas es que permite una considerable reducción de tiempo y costo; se puede obtener rápidamente la evidencia de las pruebas.

En el mercado existe una gran cantidad de software orientado a la automatización de pruebas de software; se tiene desde herramientas con costo hasta *open source*. La diferencia entre las herramientas radica en la cantidad de datos u operaciones por ejecutar. Asimismo, el tipo de configuraciones para las conexiones físicas. Estas características pueden limitar el nivel de pruebas y tiempo de ejecución.

Así como existe una diversidad de herramientas orientadas a la automatización de pruebas, también se tienen herramientas para la gestión de las pruebas, que automatiza el ciclo del testing desde la planificación, asignación de recursos y casos de pruebas hasta los informes de pruebas. Estas herramientas le permiten ver en tiempo real el estatus de las pruebas. Entre las herramientas *open source* más conocidas de gestión de pruebas tenemos:

- JIRA
- Mantis
- Bugzilla Testopia
- FitNesse
- RTH (open source)
- Salome-TMF
- Squash TM
- Test Environment Toolkit
- TestLink
- Testitool
- XQual Studio
- Radi-testdir
- Data Generator

A continuación, mencionaremos las herramientas para pruebas de automatización *open source* más uitilizadas en el mercado.

5.1. Selenium

Utilizada para la ejecución de pruebas funcionales, contiene el Selenium IDE (Integrated Development Environment), que se implementa como extensión de Firefox (genera el entorno de desarrollo y permite crear los casos de prueba para sistemas web) y el SeleniumWebDriver que ejecuta las pruebas. Es empleada para los siguientes navegadores: Google Chrome, Internet Explorer 7, 8, 9, 10 y 11 (con Windows Vista, Windows 7, Windows 8 y Windows 8.1), Firefox, Safari, Ópera, HtmlUnit, Phantomjs, Android (con Selendroid o appium)e iOS (con ios-driver o appium).

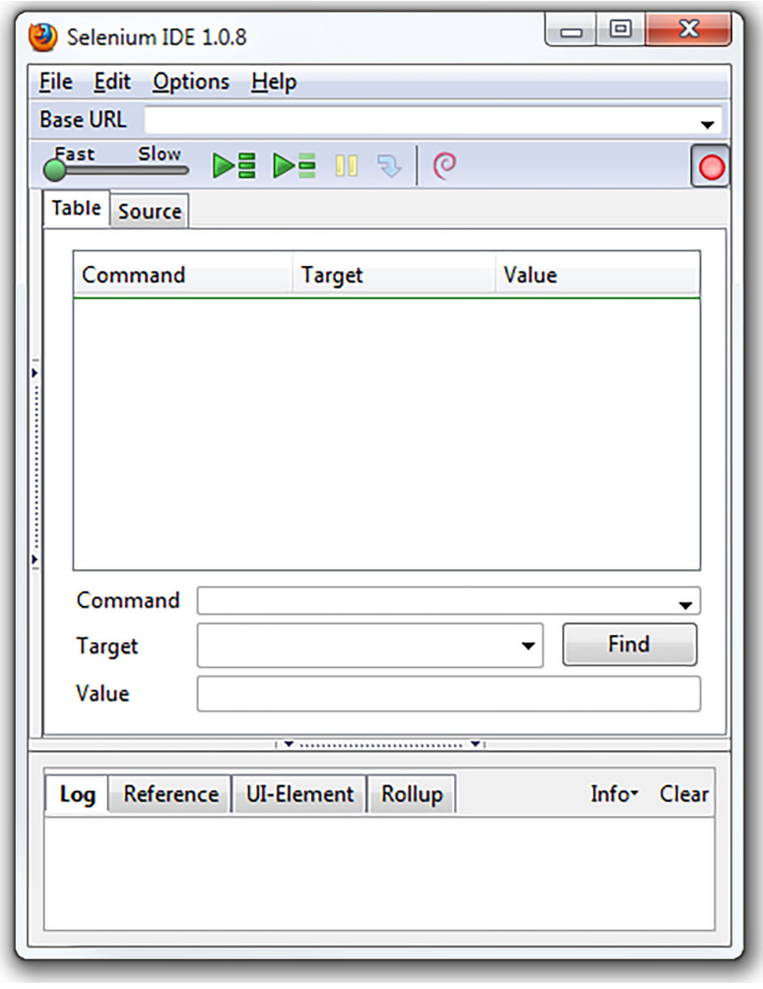


Figura 41: Pantalla de carga de casos de prueba – Selenium.
Disponible en http://bit.ly/2kB1a8X

En la figura anterior se muestra la pantalla principal del Selenium-IDE que muestra un menú para la creación de casos de prueba, así como la ventana de comandos para la edición de secuencia. A continuación, se muestra una pantalla en donde se coloca el URL de la página por probar; esto permite las pruebas desde diferentes dominios. Para el ejemplo se muestra los dominios http://news.portal.com y http://beta.news.portal.com. Selenium- IDE crea URL absolutas; para este caso con el comando open.

El caso de prueba que se ejecutará a continuación será en http://news.portal.com/about.html

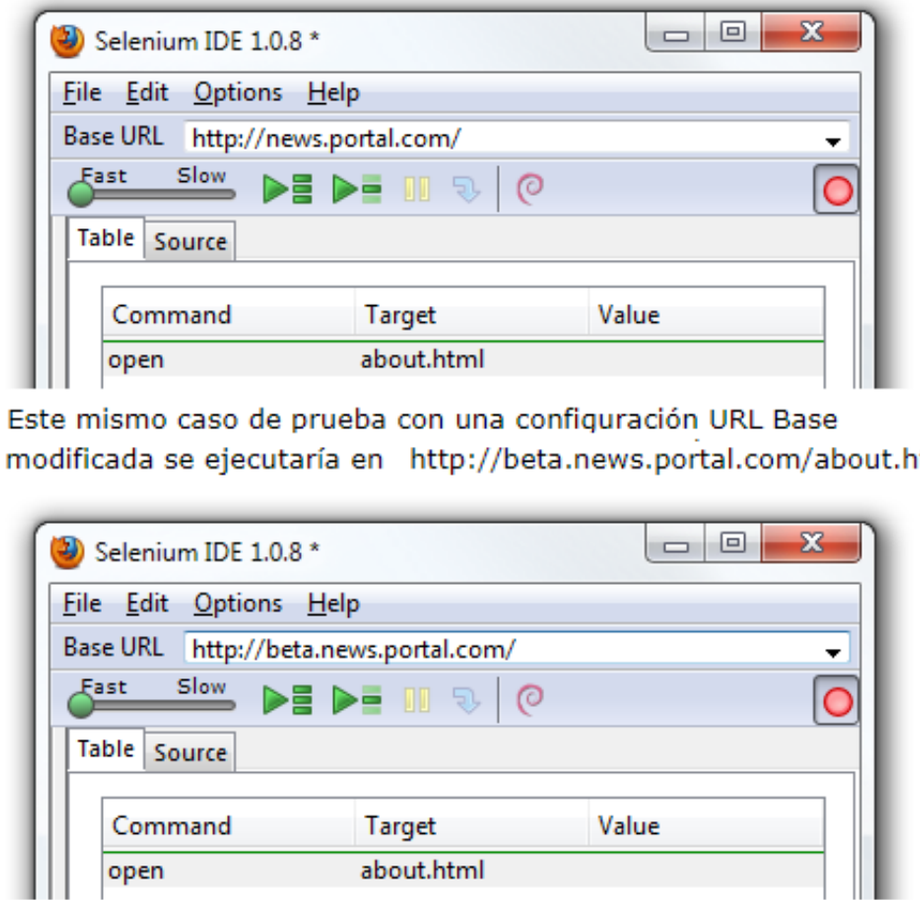


Figura 42: Ejemplo de uso de URL base para ejecutar casos de prueba.
Disponible en http://bit.ly/2kB1a8X

5.2. JMeter

Es una herramienta *open source* elaborada en java, utilizada para la ejecución de pruebas funcionales para sistemas web.

Esta herramienta configura el comportamiento de los requisitos funcionales y mide el rendimiento en recursos estáticos y dinámicos.

Trabaja con los protocolos HTTP, HTTPS, SOAP, JDBC, LDAP, JMS, Mail – POP3(S) y IMAP(S).

En el siguiente ejemplo, la pantalla de validación de *script* muestra la opción Validate, con la que se validan los *script* antes de la ejecución de los casos de prueba. Cabe mencionar que para el ejemplo se valida un grupo de *script* con una secuencia definida; los resultados podrán ser vistos en conjunto al finalizar las validaciones.

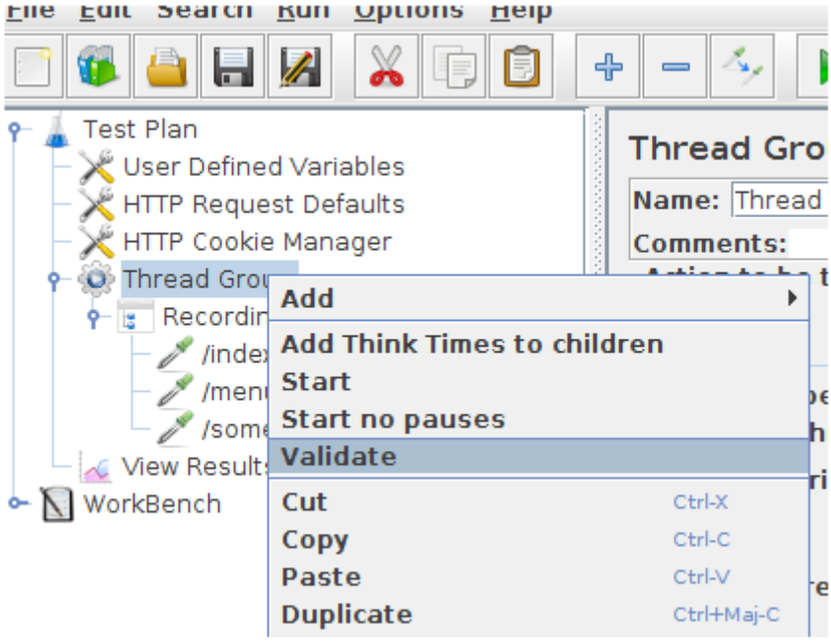


Figura 43: Ejemplo de Pantalla Validación de *Script*. Disponible en http://bit.ly/2y9slaj

JMeter también le da la posibilidad de generar un informe de la prueba ejecutada, la misma que deberá ser configurada antes de la ejecución. A continuación se muestra un ejemplo del informe de pruebas de la herramienta, en donde se visualizará el estatus de las pruebas, los casos de prueba ejecutados, el porcentaje de error. Si es que se efectúa una clasificación de errores, el mismo también podrá ser visualizado, el archivo puede ser visto en html o puede ser descargado desde la herramienta.

Summary Report

Name: Summary Report

Comments:

Write results to file / Read from file

Filename Browse... **Log/Display Only:** Errors Successes **Configure**

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB...	Sent KB/sec	Avg. Bytes
/index.html	500	5	1	103	5,82	0,00%	217,4/sec	2649,50	38,85	12480,2
/menu.html	500	5	1	31	4,01	0,00%	227,4/sec	2771,19	40,63	12480,2
/something/...	500	5	1	37	4,29	0,00%	227,7/sec	2774,97	40,69	12480,2
TOTAL	1500	5	1	103	4,77	0,00%	651,0/sec	7934,69	116,35	12480,2

Figura 44. Ejemplo de informe de pruebas. Disponible en http://bit.ly/2y9slaj

5.3. Testlink

Utilizada para la gestión de pruebas, permite la generación de casos de prueba, el seguimiento de las pruebas, trazabilidad de requisitos y generación de informes de pruebas. Se puede integrar con otros sistemas de gestión de pruebas tales como JIRA, Mantis y Bugzilla. Se puede utilizar para pruebas manuales y automatizadas.

A continuación se muestra una pantalla de la ejecución de una prueba y sus resultados:

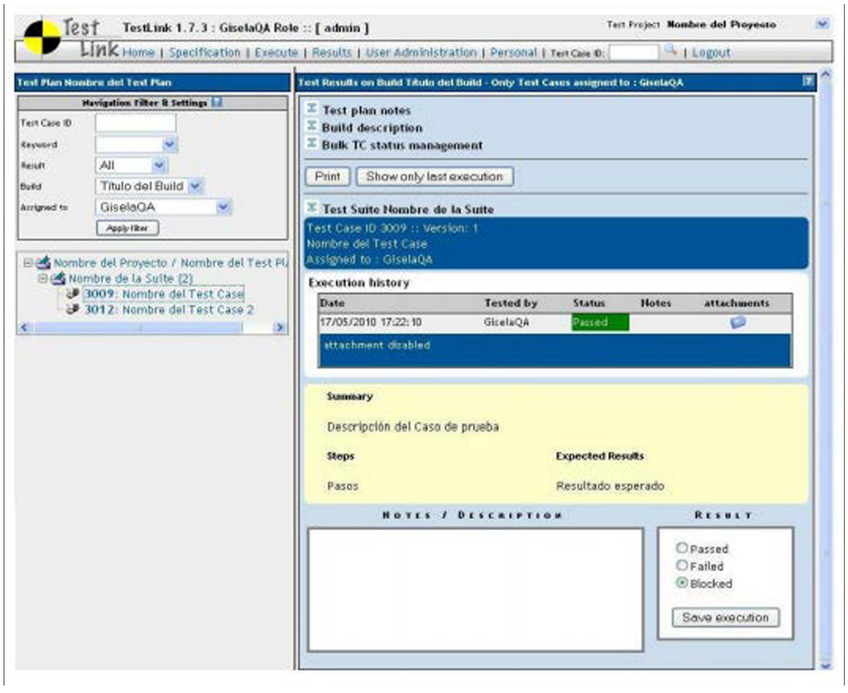


Figura 45: Ejemplo de resultado de ejecución de pruebas. Fuente: QAustral, 2010.

Uno de los beneficios que tiene esta herramienta es que permite efectuar la importación y exportación de casos de prueba. A continuación se muestra la pantalla de importación de un archivo XML.

Figura 46: Importación de casos de prueba. Fuente: QAustral, 2010.

5.4. PMD

Utilizada para las pruebas unitarias y de caja blanca, es un analizador estático del código de programación que mediante un conjunto de reglas identifica fallas en el sistema, como código duplicado o código no utilizado (variables, métodos, parámetros). Trabaja con Java, JavaScript, PLSQL, XML, XSL, Apache Velocity.

5.5. Check Style

Utilizada para las pruebas unitarias y de caja blanca, analiza el código de programación Java con base en un estándar de programación definido. Es una herramienta configurable e ideal para aquellas organizaciones que buscan el cumplimiento de estándares de codificación. Check Style proporciona ejemplos de archivos Sun Code Conventions y Google Java Style.

5.6. Sonarqube

Es una herramienta *open source* utilizada para las pruebas unitarias y de caja blanca; es un analizador de código que permite detectar fallas de código en la aplicación, configurar las métricas del proyecto y visualizar indicadores. Trabaja principalmente con JAVA.

Figura 47: Ejemplo de análisis de código fuente. Disponible en http://bit.ly/2krLfZx

En la figura anterior se muestra un ejemplo del análisis de código. Cabe mencionar que el analizador sigue los parámetros de revisión según la configuración de estándares del proyecto; el análisis que se efectúa puede ser estático o dinámico.

Sonarqube también permite gestionar y generar métricas, que le da una mayor flexibilidad para el manejo de indicadores del proyecto.

A continuación se muestra la pantalla mediante la cual se podrán efectuar modificaciones de las medidas especificadas:

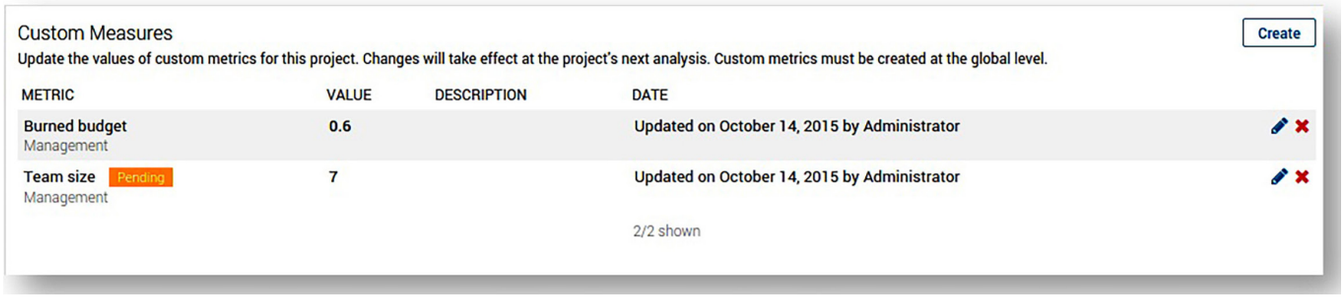


Figura 48: Ejemplo de modificación de medidas. Disponible en http://bit.ly/2xrVGOV

5.7. Bugzilla

Es una herramienta *open source* utilizada para la gestión de pruebas, muestra el seguimiento de defectos y cambios en el código. Puede instalarse en los sistemas operativos MAC, Linux y Windows.

5.8. Appium

Es una herramienta *open source* que permite la automatización de pruebas de aplicaciones móviles (Iphone y Android) utilizando WebDriver, aplicaciones web nativas (escritas con los SDK de IOS, Android y Windows), aplicaciones híbridas (Aplicaciones desarrolladas en tecnología web integrada a contenedores nativos). Prueba aplicaciones en diferentes plataformas haciendo uso de la misma API, que hace posible la reutilización entre los códigos IOS, Windows y Android.

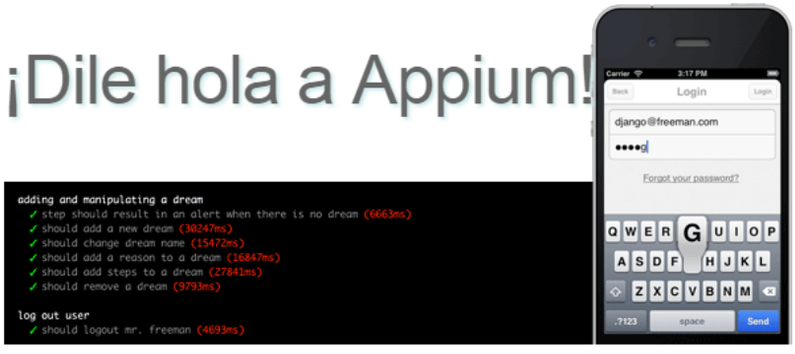


Figura 49: Código de Appium en aplicaciones nativas. Disponible en http://appium.io/

5.9. Mantis

Es una herramienta *open source* que contiene las siguientes características:

- Permite la gestión de proyectos y la gestión del ciclo de pruebas, mediante la generación de tickets de atención.
- Permite el seguimiento de fallas durante el proceso de pruebas.
- Permite la configuración de roles (programador, analista de sistemas, analista de calidad, coordinador, jefe, entre otros).
- Permite la configuración del flujo de trabajo personalizado de acuerdo con el esquema de trabajo de la organización.
- Envía notificaciones mediante correo electrónico.

- Emite reportes del seguimiento y puede catalogar los requerimientos en abierto, en proceso, devuelto, entre otros.
  Mantis es muy fácil de usar e instalar y ha sido desarrollado en PHP, con soporte para bases de datos MySQL, MS SQL y PostgreSQL, Oracle.

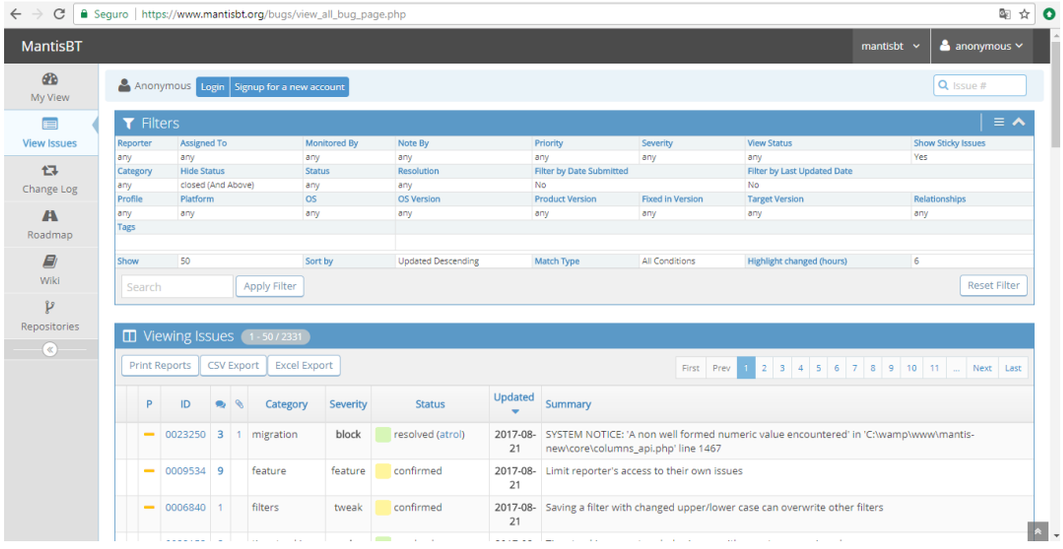


Figura 50: Pantalla Mantis. Disponible en https://www.mantisbt.org

5.10. Jenkins

Es una herramienta *open source* que verifica el código y lo automatiza, muestra rastros del código de programación y administra los cambios de versiones. Jenkins puede ser instalado en estaciones de trabajo, así como en servidores conectados a una red pública, ofreciendo opciones de configuraciones de seguridad. Jenkins es un motor de automatización, ofrece Jenkins Pipeline para dar soporte a la integración continua que va desde casos de uso hasta la entrega completa del software; para ello programa una serie de tareas en su configuración.

El diagrama de flujo Pipeline que se muestra a continuación es un ejemplo de un escenario de entrega continua modelado fácilmente en Jenkins Pipeline.

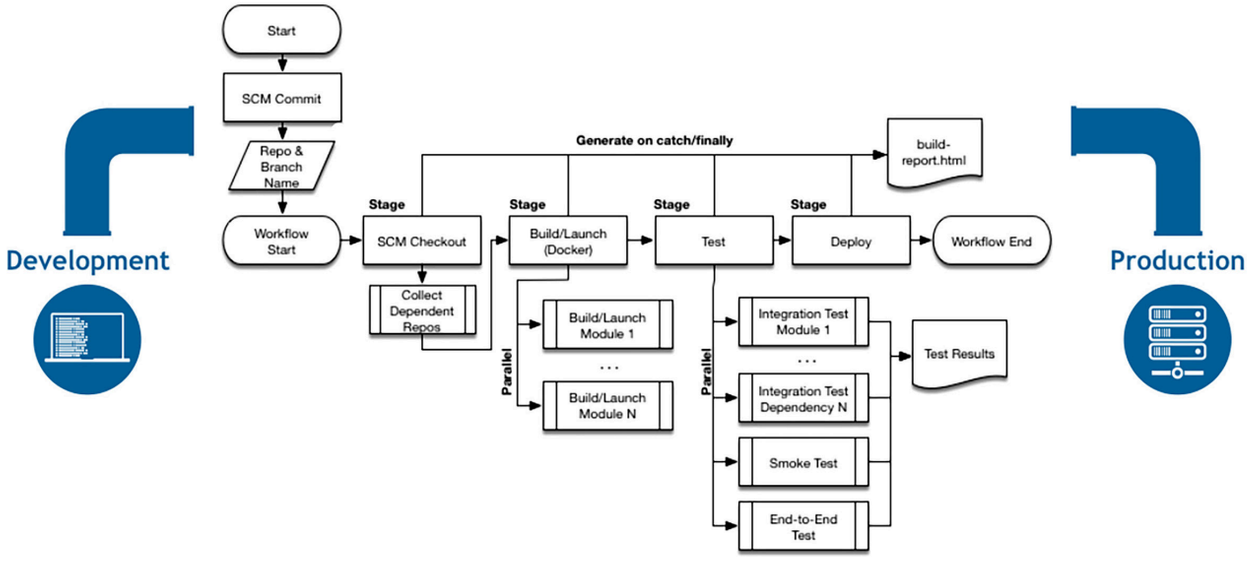


Figura 51: Flujo Pipeline. Disponible en https://jenkins.io/doc/book/pipeline/

6. Ciclo de vida de desarrollo con el uso de herramientas

Las herramientas de automatización ayudan en el desarrollo de los sistemas, y agilizan el ciclo del trabajo del desarrollo del software, por lo que hoy en día el concepto ágil ha cobrado mucha relevancia, pues las organizaciones requieren que las tecnologías vayan acorde con el desarrollo del negocio; por lo tanto, es necesario pensar en un entorno de pruebas ágil que considere repositorios comunes para el proyecto tanto para la documentación de requisitos como para las fuentes de los desarrollos de software, entornos de pruebas automatizados y servicios de integración continua. Tal como se ha visto en el ítem anterior, muchas grandes organizaciones están invirtiendo en la generación de software que soporta esta nueva tendencia del mercado; sin embargo, no se debe dejar de considerar también el hardware que soporta la misma, por lo que al momento de evaluar cualquier software hay que tener en cuenta los riesgos de implementación, integración de herramientas y hardware asociado.

Entonces, si se considera un proyecto con manejo de herramientas se podría proponer para la administración del proyecto, seguimiento del proyecto y gestión de indicadores, herramientas con JIRA y TestLink, tanto para la gestión como para el seguimiento de la operación del ciclo de vida. Para la fase de construcción, en la compilación de fuentes y generación de ejecutables se puede trabajar con Gradle, Ant o Maven. Asimismo con Jfrog Artifactory para la gestión de fuentes, permitiendo a los programadores el seguimiento de artefactos completos y automatizables desde la construcción hasta la liberación. Para el proceso de pruebas se puede trabajar con Sonarqube, JMeter y Selenium, tanto para la administración de pruebas, manejo de indicadores, pruebas de calidad y unitarias. Estas herramientas, según el tipo de pruebas, también podrían integrarse con herramientas adicionales. Para el despliegue de aplicación se puede trabajar con Jenkins. Como repositorios de documentación se propone el Subversión con almacenamiento de información en la nube o en servidores locales.

A continuación, se muestra un ciclo de desarrollo de software con manejo de herramientas.

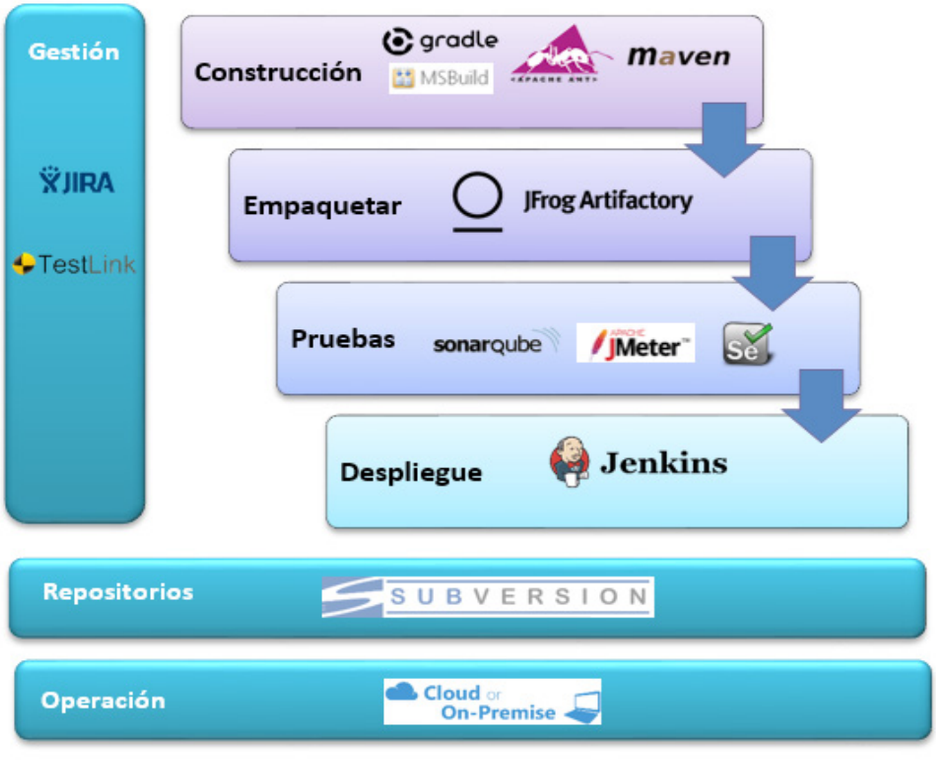


Figura 51: Ejemplo de ciclo de vida con manejo de herramientas. Fuente: Wong, 2017.

Análisis de la interacción de requerimientos

Tema n.° 5

Los requerimientos surgen de la necesidad del negocio y la evolución del mismo por ser más competitivos; por ello, la tecnología cumple un rol muy importante en el desarrollo de los negocios. La gestión de demanda es fundamental dentro del ciclo de vida del desarrollo de software debido a que se encarga de la captura de requerimientos del negocio. En ese sentido, la comprensión del negocio es un elemento crítico para la elaboración del sistema.

1. Definición de requerimiento

Un requerimiento es una noción, característica o descripción de algo que el usuario concibe para ser ejecutado por un sistema con el objetivo de optimizar su proceso de negocio. Los requerimientos primero son evaluados, se evalúa su factibilidad y luego se convierten en requisitos funcionales y no funcionales, lo cual servirá como base para el desarrollo del sistema.

Figura 52: Concepción de un requerimiento. Fuente: Wong, 2017.

2. Análisis de factibilidad de un requerimiento

La factibilidad es la evaluación previa que se efectúa para determinar si un requerimiento será o no elaborado; muestra las restricciones a nivel tecnológico, funcional, de costos y de tiempo. Muchas veces los estudios de factibilidad pueden incluir pruebas de concepto con la finalidad de evaluar si la tecnología especificada es consistente y el sistema podrá ejecutarse sin problemas.

Antes de ejecutar el estudio de factibilidad es necesario una comprensión adecuada de la necesidad del usuario; por ello, pueden darse muchas reuniones con el usuario con la finalidad de especificar adecuadamente su solicitud.

Una vez determinada la ejecución del requerimiento, se procederá a establecer las características de hardware y software requeridos para el desarrollo del sistema. Asimismo, se efectuará la planificación en costo y plazo y se determinará la metodología del ciclo de vida por seguir para su ejecución.

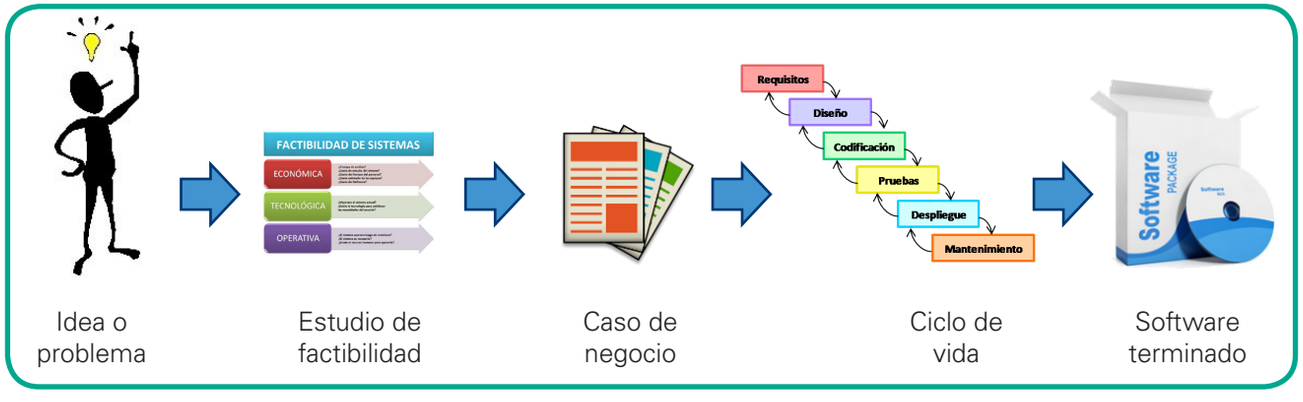


Figura 53: Ciclo de vida del producto. Fuente: Wong, 2017.

3. Interacción de requerimientos

Posterior al estudio de factibilidad se genera el caso de negocio y, en consecuencia, el requerimiento, que iniciará su desarrollo con el equipo de sistemas. En ese sentido, se define primero el ciclo de vida para su atención y las estrategias por seguir. La planificación define las reuniones necesarias que se tendrán con el usuario final para la definición a detalle de los requerimientos e iniciar la elaboración de los documentos de análisis y diseño.

Análisis formal requerimientos

Tema n.° 6

El análisis formal de requerimientos define la interacción formal de la atención del requerimiento de software entre el usuario final y el desarrollador, mediante la cual se garantiza la veracidad de los requisitos. El flujo de atención determina los hitos de aprobación de los requerimientos de manera gradual, de modo que el usuario final irá validando y aprobando la documentación y el producto por fases.

1. Características de la especificación de requisitos de software

La EEE Std 830 (1998) define como principales características de los requisitos de software a los siguientes:

- Correcta: cuando la aplicación debe cumplir obligatoriamente el requisito especificado.
- No ambigua: los requisitos definidos tienen una sola interpretación y todos los involucrados lo entienden de la misma forma.
- Completa: se considera de esta forma cuando cumple con lo siguiente:
    - Todos los requisitos significativos, relacionados con funcionalidad, desempeño, restricciones de diseño, atributos o interfaces externas deben ser reconocidos y analizados.
    - Se tienen definidas las respuestas del aplicativo a todos los tipos posibles de clases de datos de entrada en todas las situaciones. Se consideran como datos de entrada tanto a los valores válidos como inválidos.
    - Las etiquetas y referencias se encuentran completas en todas las figuras, tablas y diagramas generados en la especificación de requerimientos. Asimismo, la definición de todos los términos y unidades de medida.
- Consistente: cuando el detalle de la especificación es coherente y no tiene contradicciones en su contenido.
- Calificada de acuerdo con la importancia y/o estabilidad: se define un identificador en los requisitos según su importancia y estabilidad. Esto permite una priorización en la atención.
- Verificable: cada requisito puede ser probado de manera explícita y de esta forma verificar el cumplimiento del desarrollo versus lo requerido por el usuario.
- Modificable: si al modificar o variar el requisito la estructura se mantiene consistente.
- Rastreable: existe una trazabilidad de requisitos y los mismos pueden continuar a través del código y documentos posteriores (p. 008).

2. Análisis formal

El análisis es un documento que traslada los requisitos de negocio en requisitos de sistemas; por tanto, se especifica claramente lo que el sistema debe hacer, mediante descripciones y diagramas; los más conocidos son los diagramas UML, en cada iteración de acuerdo con el ciclo de vida el usuario irá validando cada uno de los documentos, de modo que se va avanzando en el desarrollo con mayor certeza. Hoy en día, con los métodos ágiles al usuario no solo se le muestran las especificaciones fun-

cionales y no funcionales a nivel de documentación sino que también se le va mostrando parte del producto de software ya desarrollado; de esta manera, la validación es más completa, y cualquier cambio requerido u observación se trabaja en una nueva iteración. Asimismo, la documentación es más ligera enfocándose sobre todo en el desarrollo.

En ambos casos, lo que se busca es la validación del usuario final, las aprobaciones se enmarcan en hitos para formalizar las atenciones del desarrollo, y si es que se dan cambios en el camino por una definición mal definida, esta puede negociarse mediante una solicitud de cambio.

Lectura seleccionada n.º 8

Müller, T., & Friedenberg, D. (2011). *Certified Tester Foundation Level Syllabus. Version 2011*. Disponible en http://bit.ly/2fvGrib

Actividad n.º 8

Foro de discusión sobre técnicas de pruebas.

Instrucciones:

- Ingrese al foro y participe con comentarios críticos y analíticos del tema modelos de sistemas.
- Lea y analice los temas 4, 5 y 6 del manual.
- Responda en el foro a las preguntas acerca de las pruebas.
- Mencione cuatro técnicas de pruebas por utilizar en un sistema de aplicación móvil.

Tarea académica n.° 4

I. Trabajo de investigación

a. Un *paper* sobre una herramienta de automatización, aplicada al ciclo de vida del proyecto de software.
   i. El *paper* debe tener toda la estructura correcta y referir las fuentes de manera adecuada según la normativa APA vigente.
   ii. Los puntos mínimos que se deben explicar en el *paper* son los siguientes:
      1. Descripción de la herramienta
      2. Ventajas
      3. Desventajas
      4. Cuadro comparativo de la herramienta con dos herramientas adicionales del mismo rubro (considerar costos, funcionalidad común, funcionalidad diferente)
      5. Conclusiones y recomendaciones

b. Un video subido a Youtube donde se explique lo siguiente:
   iii. Descargar versión *open source* o licenciada.
   iv. Instalación y configuración de la herramienta.
   v. Realizar una prueba de ejecución de la aplicación de la herramienta (el ejemplo debe ser efectuado por el estudiante).

II. Rúbrica

n.°	Rúbrica	Descripción	Min. puntaje	Máx. puntaje
1	Documento	El trabajo escrito debe presentar todas las pautas expuestas en formato *paper*.	0	8
2	Video	Archivo de texto donde se encuentra la URL del video en Youtube en el que se explica la diapositiva presentada.	0	12

III. Indicaciones

- Todos los puntos de la rúbrica deberán ser subidos a la plataforma en formato ZIP y como nombre del archivo deberán tener el siguiente formato: **APELLIDO_NOMBRE.zip**
- En el caso de plagio, el trabajo será invalidado y la nota será de 00.
- No referenciar adecuadamente según la norma APA vigente le restará un punto por cada referencia no establecida.

Glosario de la Unidad IV

C

Cobertura de las pruebas: Grado en que los casos de prueba prueban los requerimientos del sistema o producto de software (Indecopi, 2006).

H

Herramienta gestión de requisitos: Una herramienta de software que almacena información de requisitos en una base de datos, captura atributos de requisitos y asociaciones, y facilita el reporte de requerimientos (IIBA, 2009).

I

Inspección: Un examen visual de un producto de software para detectar e identificar anomalías de software, incluyendo errores y desviaciones de las normas y especificaciones. Las inspecciones son exámenes de pares liderados por facilitadores imparciales capacitados en técnicas de inspección. La determinación de la acción correctiva o de investigación para una anomalía es un elemento obligatorio de una inspección de software, aunque la solución no debe determinarse en la reunión de inspección (IEEE Computer Society, 1998).

R

Revisión: Proceso o reunión durante la cual se presenta un producto de software al personal del proyecto, gerentes, usuarios, clientes, representantes de usuarios u otras partes interesadas para comentarios o aprobación (IEEE Computer Society, 1998).

M

Modelado ágil: Un enfoque de desarrollo de sistemas que tiene valores, principios, prácticas útiles para los analistas de sistemas que requieran implementar un proyecto flexible, interactivo y participativo (Kendall & Kendall, 2006).

S

Software de código abierto (OSS, por sus siglas en inglés: *open source software*): Un modelo de desarrollo y filosofía de distribución de software libre y publicando su código fuente, que luego puede ser estudiado, compartido y modificado por usuarios y programadores. El sistema operativo Linux es un ejemplo (Kendall & Kendall, 2006).

Bibliografía de la Unidad IV

Appium. (2017). *Automatización para aplicaciones móviles*. Recuperado de http://appium.io

Bugzilla. (2017). *Bugzilla*. Recuperado de https://www.bugzilla.org

Checkstyle. (2017). *Checkstyle*. Recuperado de http://checkstyle.sourceforge.net/

Free Software Foundation. (3 de noviembre del 2008). GNU Free Documentation License. Version 1.3. Recuperado de http://www.gnu.org/licenses/fdl.txt

International Institute o Business Análisis [IIBA]. (2009). *A Guide to the Business Analysis Body of Knowledge: BABOK*. Recuperado de http://bit.ly/1GEPk2m

Jenkins. (2017). *Jenkins. Build great things at any scale*. Recuperado de https://jenkins.io

Kendall, K., & Kendall, J. (2013). *Systems analysis and design* (9a ed.). New Yersey: Prentice Hall.

MantisBT Team. (2017). MantisBT makes collaboration with team members & clients easy, fast, and profesional. *Mantis Bug Tracker*. Recuperado de https://www.mantisbt.org

Müller, T., & Friedenberg, D. (2011). *Certified Tester Foundation Level Syllabus. Version 2011*. Disponible en http://bit.ly/2fvGrib

Pressman, R. (2005). *Ingeniería del Software. Un enfoque práctico* (6a ed.). D.F., México:McGraw-Hill.

Sommerville, Ian. (2011). *Software Engineering* (9a ed.). Madrid: Pearson Education.

Sonarqube. (2017). *The leading product for Continuous Code Quality*. Recuperado de https://www.sonarqube.org/

TestLink. (2017). *TestLink Open Source Test Management* Recuperado de http://testlink.org

The Apache Software Foundation. (2017). *Apache JMeter*. Recuperado de http://jmeter.apache.org

Wiegers, K., & Beatty, J. (2013). *Software Requirements* (3a ed.). Washington: Microsoft Press. Disponible en http://bit.ly/2ghNk6M

Autoevaluación n.º 4

1. Las pruebas de caja blanca son una técnica utilizada en:
   a) Pruebas de aceptación
   b) Pruebas unitarias
   c) Análisis de requerimiento
   d) Diseño de requerimiento
   e) Pruebas no funcionales

2. Las pruebas de aceptación las ejecuta:
   a) Sponsor del proyecto
   b) Analista de sistemas
   c) Usuario final
   d) Programador
   e) Analista de calidad

3. Las pruebas de caja negra validan:
   a) Análisis del requerimiento
   b) Plan de pruebas del requerimiento
   c) Arquitectura del software
   d) Estructura interna del software
   e) Estructura externa del software

4. El Subversión es:
   a) Herramienta que compila de fuentes
   b) Herramienta de pruebas de código
   c) Herramienta de repositorio de fuentes y documentos
   d) Herramienta de pruebas funcionales
   e) Herramienta de gestión

5. Es una característica de la especificación de requisito de software:
   a) La especificación de no deber ser ambigua
   b) La especificación debe ser ambigua
   c) La especificación debe tener iteraciones constantes
   d) La especificación debe ser descrita por el analista de sistemas
   e) La especificación debe ser descrita por el programador

6. Después de concebirse una idea o problema se tiene que efectuar:
   a) Análisis del sistema
   b) Caso de negocio
   c) Ejecutar el ciclo de vida
   d) Diseño del sistema
   e) Estudio de factibilidad

7. Un requerimiento es:
   a) Una noción, característica o descripción de algo que el usuario concibe sea una idea.
   b) Una noción, característica o descripción de algo que el analista de sistemas concibe como sistema.
   c) Una noción, característica o descripción de algo que el usuario concibe sea ejecutado por un sistema con el objetivo de optimizar su proceso de negocio.
   d) Una noción, característica o descripción de algo que el usuario quiere para mejorar su negocio.
   e) Una noción, característica o descripción de algo que una empresa necesita para ganar dinero.

8. La actividad experiencia de usuario muestra al usuario final:
   a) El análisis de requerimientos
   b) El diseño del sistema
   c) Una metodología de requerimientos
   d) Una herramienta de gestión de proyecto
   e) Prototipos

9. La validación de la arquitectura se da en la siguiente prueba:
   a) Prueba unitaria
   b) Prueba de aceptación de diseño
   c) Prueba de integración
   d) Prueba de requisitos no funcionales
   e) Prueba de requisitos funcionales

10. El JIRA es:
    a) Herramienta de gestión
    b) Herramienta que compila de fuentes
    c) Herramienta de pruebas de código
    d) Herramienta de repositorio de fuentes y documentos
    e) Herramienta de pruebas funcionales

Anexos

UNIDAD I

Número	Respuestas
1	b
2	a
3	d
4	c
5	a
6	c
7	a
8	d
9	b
10	b

UNIDAD II

Número	Respuestas
1	e
2	b
3	b
4	c
5	a
6	c
7	c
8	a
9	b
10	e

UNIDAD III

Número	Respuestas
1	a
2	d
3	c
4	d
5	b
6	a
7	c
8	b
9	e
10	a

UNIDAD IV

Número	Respuestas
1	b
2	c
3	e
4	c
5	a
6	e
7	c
8	e
9	b
10	a

Huancayo
Av. San Carlos 1980 - Huancayo
Teléfono: 064 - 481430

Lima
Jr. Junín 355 - Miraflores
Teléfono: 01 - 2132760

Cusco
Av. Collasuyo S/N Urb. Manuel Prado - Cusco
Teléfono: 084 - 480070

Arequipa
Calle Alfonso Ugarte 607 - Yanahuara
Oficina administrativa: Calle San José 308 2° piso - Cercado
Teléfono: 054 - 412030