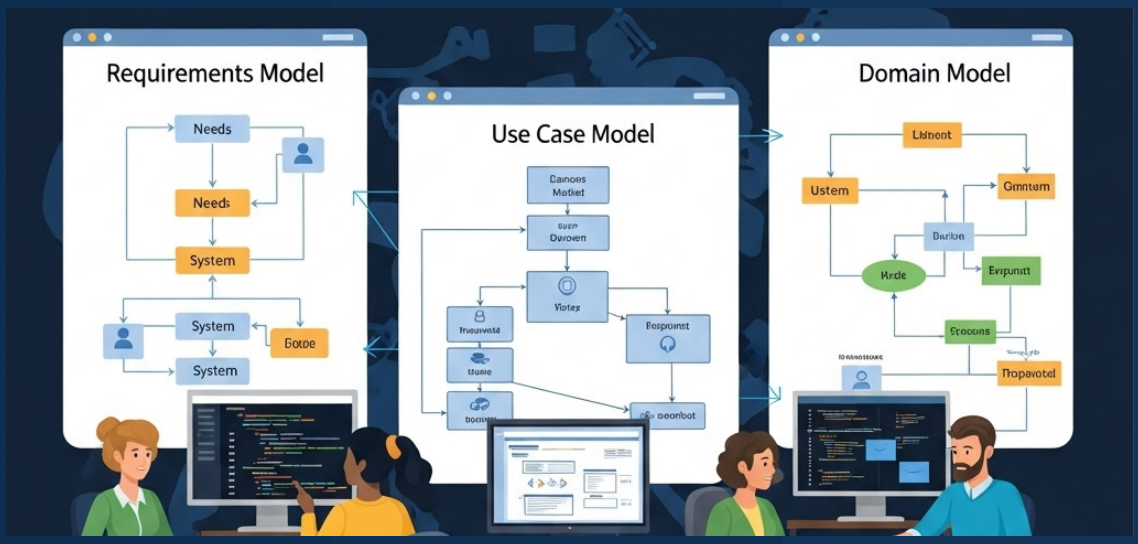
Universidad Nacional de Loja

1859

Análisis y diseño preliminar de software basado en el modelado de requisitos y casos de uso. Un enfoque teórico y práctico



Wilman Patricio Chamba Zaragocín
Edwin René Guamán Quinche
Francisco Javier Alvarez Pineda
José Oswaldo Guamán Quinche
Gerson Benjamín Zaragocín Arrobo
Georgina Ángela Valarezo Benítez

Análisis y diseño preliminar de software basado en el modelado de requisitos y casos de uso.

Un enfoque teórico y práctico

Análisis y diseño preliminar de software basado en el modelado de requisitos y casos de uso.
Un enfoque teórico y práctico

Wilman Patricio Chamba Zaragocín
Edwin René Guamán Quinche
Francisco Javier Alvarez Pineda
José Oswaldo Guamán Quinche
Gerson Benjamín Zaragocín Arrobo
Georgina Ángela Valarezo Benítez

UNL

Nikolay Aguirre, Ph.D
Rector UNL

Elvia Zhapa Amay, Ph.D
Vicerrectora Académica

Max Encalada, Ph.D
Director de Investigación

Análisis y diseño preliminar de software basado en el modelado de requisitos y casos de uso.
Un enfoque teórico y práctico

Autores:
Wilman Patricio Chamba Zaragocín
Edwin René Guamán Quinche
Francisco Javier Alvarez Pineda
José Oswaldo Guamán Quinche
Gerson Benjamín Zaragocín Arrobo
Georgina Ángela Valarezo Benítez

Revisión Par Académico:
Ramón Alfredo Toala Dueñas, Ph.D
Carlos Luis Pinargote Navarrete, Mg.Sc

ISBN: 978-9942-671-12-7

Editorial Universitaria
Contacto: comision.editorial@unl.edu.ec
Universidad Nacional de Loja
Teléfono: +59372593550
Ciudad Universitaria Guillermo Falconí, Loja - Ecuador
www.unl.edu.ec
Marzo, 2025
Loja, Ecuador

Índice General

Prefacio

El desarrollo de software ha sido una empresa en constante evolución desde sus inicios. Desde la explosión tecnológica de los años 60 hasta la revolución digital de los 90 y más allá, hemos presenciado cómo el software se ha convertido en una columna vertebral de nuestra sociedad. Sin embargo, con esta evolución también han surgido desafíos significativos, y entre ellos destaca la necesidad crucial de entender y articular los requisitos del software de manera efectiva desde sus fases iniciales.

"Análisis y Diseño Preliminar de Software Basado en el Modelado de Requisitos y Casos de Uso" se erige como una guía esencial en este panorama en constante cambio. Este libro no solo proporciona un análisis detallado de los fundamentos teóricos del modelado de requisitos y casos de uso, sino que también ofrece una perspectiva práctica derivada de la experiencia de aquellos que han enfrentado los desafíos del desarrollo de software.

Desde la comprensión de las necesidades del cliente hasta la creación de modelos conceptuales y la validación de requisitos funcionales, cada capítulo de esta obra está cuidadosamente diseñado para proporcionar a los equipos de trabajo las herramientas y el conocimiento necesarios para enfrentar los retos del desarrollo de software con confianza y competencia.

A lo largo de estas páginas, los lectores serán guiados desde los fundamentos básicos hasta las particularidades del diseño preliminar de software, con énfasis en la aplicación práctica de conceptos teóricos a través de metodologías reconocidas como ICONIX, RUP y otras metodologías ágiles.

Que este libro sirva como guía a los lectores para que encuentren no sólo conocimiento, sino también inspiración para enfrentar los desafíos del mañana con confianza y determinación, en un campo que está en constante cambio y evolución.

¡Bienvenidos a un viaje de descubrimiento y aprendizaje en el fascinante mundo del análisis y diseño de software!

Los autores.

Loja, Noviembre 2024.

Agradecimientos

El equipo de autores agradece a todas las personas que de alguna u otra forma nos apoyaron en la finalización exitosa de este libro. Un agradecimiento muy especial a nuestras familias que mostraron una gran paciencia en el proyecto de este libro. Además, agradecemos a la Universidad Nacional de Loja por ayudarnos y apoyarnos en la publicación de este libro. Finalmente, nos gustaría agradecer las aportaciones recibidas desde nuestros equipos de trabajo y estudiantes a lo largo de todos estos años, los cuales fueron una de las principales motivaciones para escribir este libro.

W. P. Ch. Z.
R. G. Q.
F. J. A. P.
J. O. G. Q.
G. B. Z. A.
G.A.V. B.

Acrónimos

SDLC	Ciclo de vida de desarrollo de software.
TIC's	Tecnología de la información y comunicación.
SW	Software.
HW	Hardware.
CRUD	Creación (Create), lectura (Read), actualización (Update) y eliminación (Delete), que son las operaciones fundamentales utilizadas para la gestión de datos o entidades sobre las bases de datos.
RSD	Requirements specification document.
DER	Documento de especificación de requisitos.
IEEE	Institute of Electrical and Electronics Engineers.
RF	Requisito funcional.
RNF	Requisito no funcional.
UML	Lenguaje unificado para el modelado.
CU	Caso de uso.
UC	Use case.
Mockup	Representaciones completamente estáticas del diseño visual.
JAD	Técnica de elicitación(obtención) de requisitos aplicadas en sesiones grupales.

Introducción

En la época de los 60 el hardware ya no fue un problema para el desarrollo de software, debido a que su construcción crecía a pasos agigantados, esto permitió que la demanda del software iniciara su crecimiento de manera exponencial. Con la aparición del internet por la década de los 90 el software tomó aún más relevancia o importancia en el mundo, haciendo necesario que las personas de alguna manera usemos algún programa informático. En estos años, el software pasó de realizar tareas "simples" a realizar tareas cada vez más "complejas" y con ello aumento la complejidad del producto y de su construcción, esto provocó que la mayoría de ellos tenga una infinidad de problemas, entre los que destacaron: el software construido no cumplía con las necesidades para las cuales fueron concebidos, no cumplir con la planificación y costos acordados, no aplicar un proceso de desarrollo de software, fallas o errores en el software, entre otros; ocasionando que gran parte del software no sea de calidad, por lo que ha esta época se la conoció como "La crisis del software".

Obtener el modelo de requisitos es una etapa crítica dentro del proceso de desarrollo de software, ya que al realizarlo de una manera incorrecta o informal determinará el fracaso del producto, mientras que, al hacerlo siguiendo un conjunto de lineamientos formales nos asegurará en gran medida que el producto construido sea utilizable para los usuarios y que el equipo de trabajo conozca lo que debe cubrir el software cuando éste llegue a su culminación.

El modelado de requisitos y casos de uso es un conjunto de actividades de ingenio y de ardua comunicación con los stakeholders, dónde se determina cuáles son las necesidades reales para el producto de software

y las expectativas de los usuarios, facilitando la detección y resolución de problemas como además la comprensión y análisis de un sistema. Para llevar esto a cabo los analistas e ingenieros hacen uso de distintas técnicas y herramientas de modelado.

El presente texto tiene la finalidad de ser una guía teórica-práctica del uso de técnicas y modelos para los analistas, desarrolladores o cualquier persona que conforme un equipo de trabajo en el desarrollo de software para la generación del modelo de especificación de requisitos y modelado de casos de uso, el cual sintetiza una de las etapas comunes dentro de cualquier metodología de desarrollo de software como lo es el "análisis" y constituye una introducción a la etapa del diseño para así evitar el problema de crear software que no cubra las necesidades del cliente.

En la unidad uno se da a conocer en qué consiste el proceso de desarrollo de software, cuáles son las etapas generales que posee éste y porqué es importante construir un producto de software aplicando un proceso de desarrollo.

En la unidad dos explica el modelado de requisitos partiendo desde su definición, estableciendo los tipos que existen: requisitos funcionales que nos indican la funcionalidad del software y los requisitos no funcionales que indican las características como atributos de calidad o arquitectura que deberá poseer el software, además se indica la manera correcta de redactarlos para enfoques orientado a objetos y su respectiva validación, así como, la forma de documentarlos a través de un documento de especificación de requisitos (DER).

En la unidad tres se presenta un resumen de las técnicas más utilizadas por los analistas con experiencia en la elicitación de requisitos, las cuales, nos sirven para: descubrir, identificar y validar los requisitos.

En la unidad cuatro se indica cómo representar el modelo conceptual o dominio inicial de un software siguiendo el enfoque orientado a objetos, el cual trata de mostrar los conceptos y sus relaciones que intervienen en el negocio tomando en cuenta los requisitos identificados. Para obtener este modelo se presentan un conjunto de buenas prácticas que facilitarán la identificación de conceptos, atributos y relaciones para representarlos en un diagrama de clases de UML (Unified Modeling Language - Lenguaje Unificado para el Modelado de Software).

Finalmente, en la unidad cinco se expone el modelado de casos de uso, el mismo que tiene como finalidad: validar los requisitos funcionales, descubrir nuevos requisitos y verificar si todos los requisitos están cubiertos con los escenarios en los que intervendrá la aplicación, sirviendo como el diseño preliminar de la aplicación. Para obtener el modelado de casos de uso se muestran técnicas para la identificación de los casos de uso (escenarios) y actores, la forma de dibujar el diagrama de casos de uso y concluir con su especificación.

Cada unidad o capítulo fue desarrollado considerando las experiencias obtenidas por parte de los autores en la participación de proyectos grandes, medianos y pequeños de software relacionados al área de la especificación de requisitos y diseño preliminar. Estas experiencias se fusionan con la teoría y mejores prácticas en el proceso de desarrollo de software y metodologías de desarrollo de software ágiles centradas en el enfoque orientado a objetos como ICONIX, RUP (Rational Unified Process), entre otras.

1. El proceso de Desarrollo de Software.

1.1. ¿Qué es un proceso de desarrollo?

Aunque el término proceso de desarrollo puede resultar muy amplio, el que nos interesa es el asociado al área del software. El proceso de desarrollo es el conjunto de actividades que deben realizarse con el objetivo de transformar las necesidades de información que tiene un usuario en aplicaciones o programas que le permitan solucionar esas necesidades. Con esta definición, puede decirse que todo software debe cubrir las necesidades de algún tipo de usuario. Por lo general, las aplicaciones informáticas cubren necesidades de tipos de usuario muy diversos.

En el proceso de desarrollo de software es importante entender cómo el análisis debe manejar el entorno del usuario y cómo esa interacción entre ambos puede permitir un buen inicio del proceso de desarrollo. En la figura 1 se explica el proceso de desarrollo de software; el cual va desde la especificación de las necesidades de un usuario determinado, hasta la entrega conjunta del programa que permita solucionar esas necesidades.

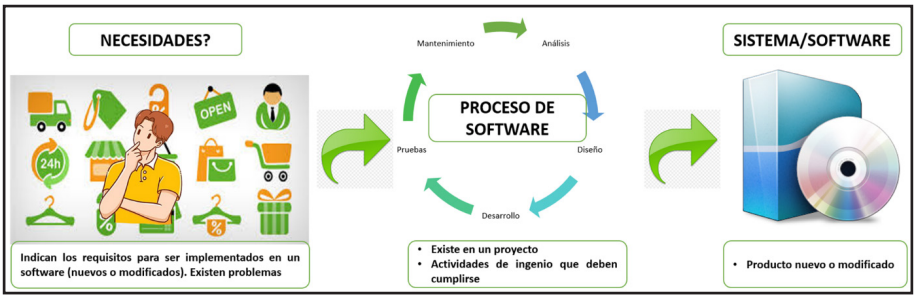


Figura 1. Esquema general del proceso de desarrollo de software / sistema.

Muchas de las aplicaciones de las que somos usuarios se han construido siguiendo el proceso de desarrollo de software; pero, por su generalidad, pueden utilizarse por usuarios de los más diversos grupos, a estas aplicaciones se las denomina software de propósito general. Otras aplicaciones se desarrollan para fines específicos, por lo que su beneficio se destina a un grupo pequeño de usuarios o negocios, en este caso se denomina software a la medida.

Sin importar cuál sea el tipo de software que se pretenda construir, la mirada debe concentrarse siempre en el usuario final, pues son sus necesidades las que determinarán las características específicas del producto de software que se debe desarrollar.

Para dar una definición más técnica acerca del proceso de desarrollo de software, se dirá que es la forma de organizar aquellas actividades que permiten la construcción de los sistemas de software.

Sin pretender ser exhaustivo en la explicación de las causas, una de ella es la falta de organización durante la construcción de la aplicación, es decir el apego nulo a un buen proceso de desarrollo.

1.2. Importancia del proceso de desarrollo de software.

Hasta ahora se ha aprendido que las necesidades de los usuarios que requieren ser solucionados construyendo software tienen dos fases: (1) la fase de resolución del problema a través del análisis y diseño; (2) la implementación a través de la codificación de los diagramas generados en el diseño.

Durante mucho tiempo, la implementación de software fue una disciplina que creció de forma desorganizada sin una metodología o pasos sistemáticos que las guíen. Su base principalmente fue la

creatividad e imaginación de cada persona involucrada en el desarrollo de un producto de software determinado.

Con el tiempo, está manera negativa de hacer software ha perdido fuerza, surgiendo una cantidad de metodologías y procesos que han permitido organizar las distintas actividades que intervienen en la construcción o desarrollo de software. Además, al irse incrementando la complejidad de los sistemas solicitados por el usuario, la necesidad de hacer uso de un proceso de desarrollo, es cada vez más imperiosa.

En el reporte CHAOS[1] emitido en el 2015 por el Standish Group International, organización que presenta un análisis de la ejecución de más de 25000 proyectos de software en el periodo 2011 al 2015, muestra que el 60 % de los proyectos no son entregados a tiempo, 56 % sobrepasan el presupuesto y 44 % no cumplen con los objetivos propuestos. También se especifica que en un 44 % el cliente no estaba satisfecho con el resultado. Como se observa en la figura 2, gran parte de los proyectos de software han fracasado, ocasionando de alguna manera una crisis en el software o la desconfianza de los clientes para apostar en la construcción del software.

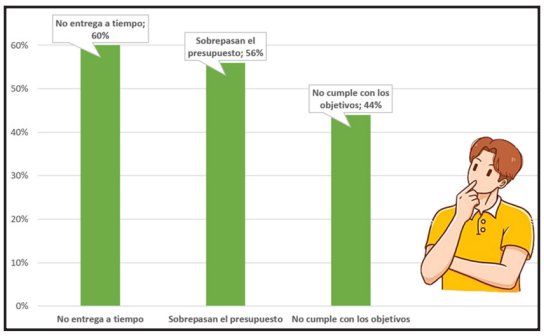


Figura 2. Análisis de proyectos de software.

1 Para analizar el reporte CHAOS revise el siguiente enlace: https://www.standishgroup.com/sample_research_files/CHAOSReport2015-Final.pdf

Las nuevas tecnologías solucionan algunos inconvenientes en el proceso de desarrollo de software, pero la tecnología no es la única que puede resolver los problemas. Lo importante es entender que el proceso de desarrollo de software es una actividad creativa, una actividad mental en esencia y una actividad que implica una gran capacidad de comunicación entre los involucrados (usuarios, analistas, desarrolladores u otras personas interesadas) del proyecto. A las personas interesadas o involucradas se las conoce como stakeholders.

La importancia de todo esto, es, aplicar un buen proceso de desarrollo, de acuerdo a lo dicho hasta ahora. En la construcción de sistemas útiles para los usuarios finales, es bueno considerar entonces qué, aunque durante algunas etapas del proceso, se puede prescindir del usuario, el pensamiento del analista siempre debe orientarse a aquello que él necesita.

1.3. Etapas del proceso de desarrollo de software.

Debido a que se ha descrito el proceso de desarrollo como un conjunto organizado de actividades, se presenta una visión general de las etapas y el objetivo que tiene cada una de ellas.

Tabla 1. Etapas del proceso de desarrollo.

Etapa	Objetivo de la etapa
Análisis de Requisitos.	Consiste en la clarificación y entendimiento por parte del analista de lo que quiere que el software haga, a través de la interacción y constante trabajo con los futuros usuarios del sistema y las partes interesadas, realizando una investigación real acerca del problema y de sus partes, sin embargo, no toma en cuenta ninguna solución. Se establecen las necesidades reales del software o producto (requisitos) de los usuarios y partes interesadas, a través del documento de especificación de requisitos (DER).

Etapa	Objetivo de la etapa
Diseño.	Permite pensar en cómo solucionar los requisitos (problemas) que se han identificado en el análisis. Establece la arquitectura donde se indica la solución lógica y física acerca del problema en cuestión.
Implementación.	Esta etapa permite la traducción de los modelos del diseño (arquitectura) o solución a un conjunto de programas y tecnología que al aplicarla permitirá cubrir las necesidades de los usuarios.
Explotación.	Durante esta etapa, el sistema entra en una fase de producción, lo que quiere decir que empieza a servir en aquellas necesidades para las cuales fue concebido.
Mantenimiento.	En esta etapa se llevan a cabo tareas relacionadas con la corrección de inconvenientes detectados, la actualización de funcionalidades existentes mediante mejoras y/o la creación de nuevas funcionalidades para el producto de software que está en producción.

En la tabla anterior, se detallan las etapas o fases que existen en un proceso de desarrollo y cada una se subdivide en actividades. A lo largo del tiempo, estas etapas no han variado y lo importante es conocer qué rol cumple cada una de ellas en el proceso de construcción del software.

La figura 3 muestra las actividades fundamentales y comunes relacionadas a cada etapa que se realizan independientemente de la elección de un proceso de desarrollo de software.

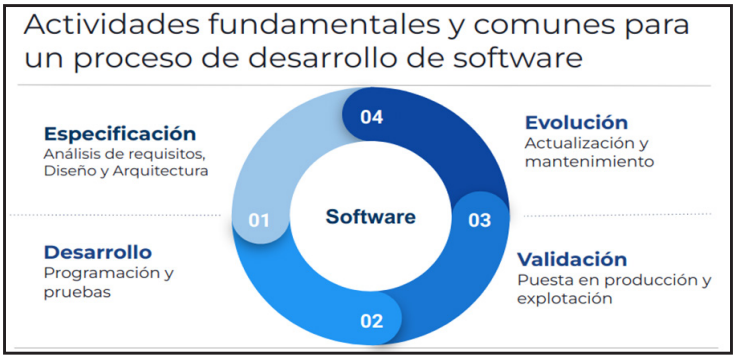


Figura 3. Actividades fundamentales y comunes para un proceso de desarrollo de software.

Tenga en cuenta que, aun cuando a las etapas se la ha citado de manera secuencial, la realidad en el desarrollo es muy diferente, pues muchas ocasiones las actividades del proceso del desarrollo son iterativas e incrementales, entonces, no es necesario terminar con una etapa para iniciar con las actividades de otra, de esta manera, pueden irse obteniendo partes del diseño del sistema.

El proceso al ser iterativo, irá descubriendo nuevas características del sistema con el avance del desarrollo, además, la característica de un proceso de ser incremental permite que con cada iteración se obtenga un refinamiento constante de aquellas características que forman parte de la descripción del sistema. También, el proceso proporciona un alto grado de seguimiento entre las necesidades del usuario inicialmente definidas y el software obtenido como resultado, siendo así durante todas las etapas del mismo.

Por ello, no debe verse al proceso como una "camisa de fuerza", en la cual deben seguirse los pasos en forma ordenada y relajada, sino que estos deben permitir lograr el objetivo primordial de contar con buenos modelos de diseños que permitan construir lo que el usuario está esperando obtener realmente.

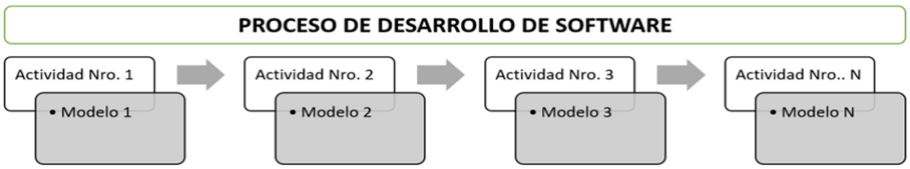


Figura 4. El objetivo del proceso de desarrollo de software es obtener modelos.

Cabe considerar que las actividades que se ejecutan dentro de un proceso de desarrollo de software, son actividades de ingenio para el equipo de trabajo y estas actividades deben de alguna manera documentarse, por lo tanto, el objetivo de la ejecución de las etapas y/o actividades para el equipo de trabajo es obtener modelos.

Entonces, ¿es necesario explicar en forma breve qué es un modelo?

Modelo es una técnica que se usa para poder describir o comprender mejor la realidad, por lo tanto, generar o crear modelos nos sirven para: explicar, documentar y como referencia para construir objetos.

No se debe caer en el error de pensar que el modelo es un diagrama o dibujo, pues no siempre es posible describir de esa forma, también puede ser una descripción en forma de texto o la combinación diagramas, dibujos o texto, lo importante es utilizar cualquiera herramienta que me permita cumplir la finalidad de explicar lo que se entendió (análisis) o lo que se pretende hacer (diseño).

Para comprender un modelo, imaginen un profesional de la arquitectura, para poder plasmar la idea de un proyecto de vivienda, él debe generar un conjunto de modelos (planos, maquetas) para poder explicar y documentar la idea de solución a su cliente indicando cómo va lucir su vivienda, cuál será la distribución, por dónde deben ir las instalaciones de agua, energía eléctrica y así muchas otras cosas más; y, por lo tanto, servirán como guía para su construcción.

Los proyectos de software son muy similares a los proyectos de vivienda, en la cual se desarrolla a través de un proceso que inicia desde las necesidades del usuario hasta la obtención del producto, pasando por un conjunto de actividades donde se deben generar modelos que me permitan establecer una comunicación adecuada con todas las partes interesadas del proyecto.

En conclusión, el proceso de desarrollo de software consiste en modelar el software, lo cual es una técnica generalizada para ayudar a los ingenieros de software a comprender, diseñar y comunicar aspectos del software a las partes apropiadas (aquellas entidades que tienen un interés declarado o implícito en el software, por ejemplo: usuario, comprador, proveedor, arquitecto, autoridad certificadora, evaluador, desarrollador, ingeniero de software y quizás otros).

2. Modelado de requisitos (Determinación de Requisitos).

El modelado de requisitos se enmarca en una de las actividades fundamentales del proceso de desarrollo de software, que es la actividad de especificación, la misma que comprende el análisis, diseño y arquitectura del software.

Entonces, el modelado de requisitos al encontrarse dentro de la fase de análisis se debe enfatizar en la investigación del problema y los requisitos antes que la solución, es decir, ¿cómo entender al cliente?

"Análisis" es un término amplio, pero al final lo que se debe realizar es lo siguiente:

- El modelado de requisitos, que consiste en la especificación de los requisitos (una investigación de las necesidades reales de los stakeholders); y/o

- El modelo del dominio, que consiste en la representación de los conceptos que intervienen en el dominio del problema (una investigación de los conceptos del mundo real de acuerdo con el contexto del problema).

Muchas de las veces los usuarios que requieren un nuevo software o soporte de un software (cambios o nuevas funcionalidades) denotan que tienen problemas y los expresan como deseos (requisitos) para que les resuelva el software, por tal razón, los analistas deben tener la capacidad de determinar cuál de esas necesidades son solamente deseos y cuáles de ellas son las necesidades reales (requisitos). Para determinar cuáles de esas necesidades son reales se debe considerar algunos aspectos tales como: el alcance del software, presupuesto, tiempo entre otros.

Por lo tanto, se puede decir que establecer las necesidades reales del sistema es la fase de determinación de requisitos a través del modelado de requisitos del proyecto de software, con los cuales de alguna manera se asegura el entendimiento del problema y el buen inicio del proyecto de software.

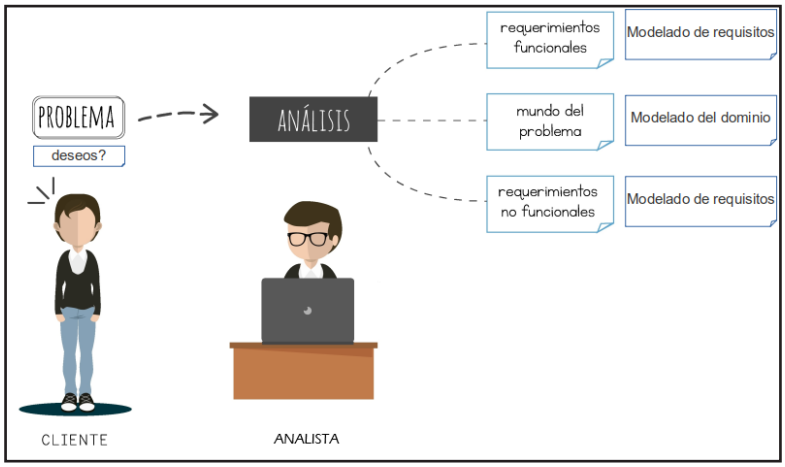


Figura 5. Etapa de Análisis. Fuente tomada de la Universidad de Salamanca – Dpto. de Informática y Automática.

Entonces, el principal objetivo del modelado de requisitos es la identificación de las necesidades reales del usuario y sin ambigüedades, de tal manera que una vez finalizado el proceso de desarrollo del sistema, no existan sorpresas en cuanto al producto construido y el producto esperado por el usuario.

Es importante destacar y aclarar que el uso exacto de estos términos (requerimientos y requisitos) puede variar en diferentes contextos y organizaciones, por lo que se recomienda consultar o establecer la terminología específica y las definiciones utilizadas en un proyecto o empresa en particular. Para nuestro contexto, de aquí en adelante, los términos requerimientos y requisitos los tomaremos como sinónimos.

Otro objetivo del modelado de requisitos es resolver el típico problema de comunicación que existe entre las partes interesadas (cualquier persona que denote interés en el sistema, como: usuario, cliente, equipo de desarrollo, etc). La siguiente imagen trata de mostrar el problema de la comunicación que suele pasar en el desarrollo de software.

Figura 6. Ejemplo de problema de comunicación.

En sí, el modelado de requisitos consiste en trabajar con los usuarios, de preferencias con los expertos o con aquellos que conozcan del dominio, los cuales, saben lo que deben hacer, y sobre todo ayudan a clarificar aquellos puntos, en los cuales, el analista tiene dudas o quieren comprender su funcionamiento, para poder tener una visión clara y el alcance del proyecto de software.

2.1. ¿Qué es un requisito o requerimiento?

Se define como una descripción detallada de las necesidades que se tiene de un producto. Según la IEEE, 1999a, un requisito es una condición o capacidad que debe tener un sistema o un componente de un sistema para satisfacer un contrato, una norma, una especificación u otro documento formal.

Un requisito hace referencia a la definición de la funcionalidad, servicio, característica o restricción que un producto de software debe cumplir para satisfacer las necesidades de los diferentes actores o stakeholders.

Además, se puede decir que un requisito es una declaración singular que contiene una estructura: actor – verbo – complemento (Hull, 2005, p. 80).

El actor es cualquier ente (persona, software o módulo) que cumple un rol autorizado ante el nuevo sistema para realizar algo sobre el mismo. El verbo es absolutamente crítico y obligatorio porque describe la función, tarea o acción que se realizará en el sistema y deberá ser declarado en presente o futuro. El complemento se encarga de completar o ampliar el significado de la acción (verbo) dentro del requisito.

En algunos requisitos, cuando el actor se omite, significa que son requisitos que el sistema directamente los realizará y por ende son ocultos al usuario.

A continuación, se indica un ejemplo de cómo mejorar la redacción de requisito funcional utilizando una estructura más detallada. En lugar de la estructura actor-verbo-complemento, se propone la estructura actor-verbo-concepto-complemento, que permite describir con mayor precisión tanto el dominio del problema y como el de la solución. Este enfoque es ilustrado en el contexto de una aplicación para la gestión de restaurantes:

“El sistema debe permitir al cliente agregar platos y bebidas a la orden de pedido”; donde: el actor es “el cliente”, el verbo “agregar” indica la acción a ejecutar sobre el concepto, el concepto “platos y bebidas” que son representaciones existentes en el negocio, y, el complemento “a la orden de pedido” que agrega una explicación adicional al verbo y al concepto.

Los requisitos deben cumplir algunas características que garanticen la calidad de los mismos. Las definiciones de algunas de las características de un buen requisito son:

- **Correcto**: es una declaración precisa de la capacidad del sistema en sus entradas, sus salidas, sus interfaces y cómo interactúa con su entorno.

- **Atómico**: se refiere a descomponer el requisito hasta su nivel lógico más bajo.

- **No ambiguo**: se refiere a que el requisito tiene que tener una solo interpretación, evitar en la redacción términos como algunos, casi, varios, muchos, etc.

- **Completo**: se encuentra toda la información necesaria para definir el actor, el verbo que describe qué acción debe hacer y el resultado de esa acción que completa la descripción de la necesidad.

- **Consistente**: cuanto más corta sea la declaración, es probable que sea más fácil de leer. Esto significa que está usando palabras más cortas, más precisas y verbos más activos.

- **Rastreable**: el estándar IEEE 830 define el atributo rastreable como la trazabilidad a un origen y a la futura documentación de desarrollo o mejora.

Los requisitos permiten definir tanto la funcionalidad esperada por el sistema, así como sus características principales, aquellos requisitos que definen la funcionalidad del sistema se los denomina “requisitos funcionales”, en tanto que aquellos que definen las características del sistema son los “requisitos no funcionales o atributos del sistema”.

El analista deberá descubrir los requisitos una vez que haya aplicado un conjunto de técnicas de elicitación o de recolección de información como: la entrevista, la observación, encuesta, lluvia de ideas, prototipos, etc.

2.2. Requisitos Funcionales.

Los requisitos funcionales son descripciones formales que expresan las funciones específicas que el producto de software debe realizar para cumplir con las necesidades de los diferentes actores o partes interesadas (stakeholders). Estas descripciones detallan qué debe hacer el software en términos de servicios, procesos y comportamientos que permitan resolver el problema o satisfacer una necesidad. Para formular estos requisitos es fundamental considerar tanto el dominio del problema como el dominio de la solución, ya que el entorno en el que se usará el software influye directamente en cómo debe operar y cumplir esas reglas de negocio. Existen tres enfoques principales para representar la funcionalidad:

1. Lista de requisitos (actor-verbo-complemento).
2. Casos de uso.
3. Historias de usuario.

Para la lista requisitos, es una enumeración de funcionalidades que se expresan detalladamente a través de un listado de oraciones siguiendo las estructuras “actor – verbo – concepto – complemento” o “actor – verbo – complemento”, incluyendo en las mismas las siguientes frases en presente o futuro: “deberá hacer”, “debe hacer”, “permite” o “permitirá”.

Los casos de uso tratan de describir estructuradamente el comportamiento de uno o varios requisitos a través de la interacción entre el actor y la aplicación en cierto escenario. Son útiles para capturar, validar y describir los requisitos funcionales del software desde la perspectiva del usuario, pero sin tomar en cuenta detalles de tecnología. Los casos de uso lo profundizaremos más adelante en el capítulo 5. “Modelamiento de Casos de Uso” como técnica para el diseño preliminar de la aplicación.

Las historias de usuario son narrativas breves de los requisitos de un cliente o usuario. Su objetivo es capturar de manera clara, simple, concisa y precisa lo que el usuario quiere lograr o alcanzar dentro del producto de software, sin entrar en detalles técnicos. Para formularlas, se sigue el siguiente patrón rol – función – beneficio, respondiendo a las siguientes preguntas: ¿Cómo? (rol) representa, al actor que requiere la nueva capacidad o funcionalidad; ¿Quiero hacer? (función) indica, la funcionalidad que el actor desea realizar; ¿Para qué? (beneficio) que es el resultado o el valor que el actor espera obtener. Además, cada historia de usuario debe incluir criterios de aceptación, que son condiciones que deben cumplirse para considerar que la historia de usuario ha sido cubierta para ese requisito. Las historias de usuario son muy utilizadas en enfoques ágiles, ayudando a los equipos de desarrollo comprender mejor lo que el actor necesita. En la tabla 2 se presenta un ejemplo de historia usuario con sus componentes básicos.

Tabla 2. Ejemplo de historia de usuario con sus componentes básicos.

Historia de usuario	Visualizar el historial de pedidos (Requisito).
ID de historia de usuario	001
Descripción	Como cliente registrado, quiero poder visualizar el historial de mis pedidos, para revisar mis consumos pasados fácilmente.
Criterios de aceptación	
1. El usuario debe poder acceder a su historial desde su perfil.	
2. El historial debe mostrar la fecha, los productos, el precio total y el sujeto a quien se le realizó el comprobante.	
3. El usuario debe poder filtrar sus pedidos por fecha y estado (entregado, en proceso, cancelado).	
4. Cada pedido en el historial debe tener un enlace para ver los detalles completos del pedido y el comprobante generado.	

Es importante señalar que tanto los enfoques de casos de uso y de historias de usuario no solo sirven para describir requisitos, sino que también pueden ser utilizados como técnicas de elicitación de requisitos. La elección de cuál emplear dependerá de la habilidad y criterio del analista, quien decidirá cómo utilizarlos o aplicarlos según las necesidades del proyecto y el contexto de los interesados.

Los requisitos funcionales están obligados a definir en sus declaraciones:

- ¿Cuáles entradas debe aceptar el sistema?
- ¿Cuáles salidas debe producir el sistema?
- ¿Qué datos debe almacenar el sistema que utilizan otros sistemas?
- ¿Qué operaciones internas debe realizar el sistema?

Existe una clasificación de tipos de requisitos funcionales (Koelsch, 2016, p. 31) que sirven como guía para categorizarlos:

- **Reglas de negocio**: deberá definir todo el tipo de información que debe crearse, leerse, actualizarse y eliminarse (también conocida como CRUD); y cualquier otra operación que necesita ser realizada.
- **Transacciones**: cubre varios aspectos de las transacciones. No sólo examinará la entrada de una transacción, sino también la modificación, eliminación, desactivación / cancelación y verificación y manejo de errores.
- **Funciones Administrativas**: describen las funciones que un administrador del sistema realiza en su sistema, por ejemplo, agregar un usuario del sistema, reactivar cuentas de usuario, etc.
- **Búsqueda e Informes:** se debe especificar qué datos puede consultar un usuario. En la mayoría de los casos, los usuarios pueden ver todo dentro de una base de datos.

2.3. Requisitos No Funcionales.

Los atributos del sistema son las características que se necesita que el sistema cumpla; sin llegar a ser funciones del sistema. Al igual que los requisitos funcionales, los atributos se irán descubriendo durante la aplicación de las técnicas de elicitación como la entrevista, lluvia de ideas, etc. Son más críticos que los funcionales: el incumplimiento de un requisito funcional degrada el software, el de un no funcional lo puede inutilizar, por ejemplo, en el caso de la disponibilidad.

Para garantizar que el sistema cumpla con lo que espera el usuario, los requisitos no funcionales deben tener en consideración los atributos, que son características de calidad del sistema; estos atributos se deben tener en cuenta al diseñar e implementar el Software. Algunos atributos que se pueden considerar son:

- Rendimiento.
- Disponibilidad.
- Seguridad.
- Accesibilidad.
- Usabilidad.
- Estabilidad.
- Portabilidad.
- Costo.
- Operatividad.
- Interoperabilidad.

De la misma manera que en los requisitos funcionales, no existe un formato ideal para representar los requisitos no funcionales. A continuación, se indica un ejemplo de redacción de requisitos no funcionales tomando en cuenta los atributos de calidad del software (Fernandes, 2016, p. 48).

Tabla 2. Redacción de requisitos no funcionales agrupados por atributos de calidad.

Código	Categoría	Descripción
RNF01	Usabilidad.	El tiempo de aprendizaje del software para un usuario deberá ser menor a 2 horas.
RNF02		La aplicación web debe poseer un diseño "Responsive" a fin de garantizar la adecuada visualización en múltiples computadores personales, dispositivos, tablets y teléfonos inteligentes.
RNF03		El sistema debe poseer interfaces gráficas amigables y bien formadas.
RNF04		La aplicación debe proporcionar mensajes de aviso y de inconvenientes que sean informativos y orientados al usuario final.
RNF05	Eficiencia.	El tiempo de respuesta para solicitudes debe ser menor a 7 segundos.
RNF06		El software debe ser capaz de operar adecuadamente con un máximo de 10.000 usuarios con sesiones concurrentes.
RNF07	Seguridad.	Los permisos de acceso al sistema podrán ser cambiados solamente por el administrador.
RNF08		Todas las comunicaciones externas entre servidores de datos, aplicación y cliente del sistema deben estar encriptadas utilizando el algoritmo RSA.
RNF09		Las credenciales del password o información relevante deben estar encriptadas utilizando el algoritmo RSA dentro del repositorio de persistencia.

Código	Categoría	Descripción
RNF10	Seguridad.	Si se identifican ataques de seguridad o fallas de seguridad del sistema, el mismo no continuará operando hasta ser desbloqueado por un administrador de seguridad.
RNF11		Todos los sistemas deben respaldarse cada 24 horas. Los respaldos deben ser almacenados en una localidad segura determinada por el administrador del sistema.
RNF12		La nueva aplicación debe desarrollarse aplicando patrones y recomendaciones de programación que incrementen la seguridad de datos.
RNF13		La aplicación debe conceder acceso a los usuarios de acuerdo a sus credenciales y permisos respectivos.
RNF14	Mantenibilidad.	El software debe soportar las nuevas funcionalidades requeridas como pedidos en línea y entrega a domicilio.
RNF15		Debe ser de fácil mantenimiento y corrección de bugs.
RNF16	Disponibilidad.	El sistema debe tener una disponibilidad al menos de un 99 % en un año.
RNF17		El tiempo para iniciar o reiniciar el sistema no podrá ser mayor a 5 minutos.
RNF18	Portabilidad.	El sistema será desarrollado para las plataformas Windows, IOS, Linux, Android.
RNF19		La nueva aplicación debe manejar fuentes del alfabeto en inglés e idiomas latinos en español.
RNF20		La interfaz de usuario será implementada para navegadores web únicamente con HTML5 y JavaScript.
RNF21		La plataforma de programación a utilizar será Flutter, Java, Pyhton, etc.
RNF22		Se utilizarán Bases de datos compatibles con la plataforma de programación y sistemas operativos.

Todos los requisitos (funcionales/no funcionales) deben ser documentados, para ello, se debe elaborar un documento de especificación de requisitos (RSD - DER), el cual, consiste en la lista de requisitos que el sistema debe cubrir. Según la IEEE (Institute of Electrical and Electronics Engineers) específica el estándar IEEE-830 para la documentación de requisitos, pero se puede adaptar cualquier otro formato que permita registrar los requisitos en forma ágil y precisa que sirva para los stakeholders. En la siguiente figura se indica el esquema del RSD o DER para especificación del estándar IEEE-83:

Figura 7. Esquema del DER para la especificación del estándar IEEE-830.

En las tablas 4 y 5 se indican ejemplos de formatos más comunes y sencillos adoptados por las empresas para ser utilizados como documentos de especificación de requisitos, éstos pueden ser utilizados como alternativa a la especificación del estándar IEEE-830.

Tabla 4 Especificación de requisitos en formato de ficha.

FICHAS DE REQUISITOS FUNCIONALES PARA ÓRDENES DE PEDIDO	
ID. Del requisito:	RF001
Nombre del requisito:	Visualizar menú
Descripción:	El sistema debe permitir a un cliente, visualizar el menú que oferta el establecimiento para la fecha (actual o futura).
Prioridad:	Alta

ID. Del requisito:	RF002
Nombre del requisito:	Buscar productos (platos / bebidas)
Descripción:	El sistema debe permitir a un cliente, buscar productos (platos / bebidas) disponibles que se encuentran en el menú a través del nombre, descripción o por precio.
Prioridad:	Alta

Tabla 5 Especificación de requisitos en formato de listado.

REQUISITOS FUNCIONALES PARA ÓRDENES DE PEDIDO	
El sistema debe permitir:	
Código	**Descripción**
RF001	Al cliente, visualizar el menú disponible en el día.
RF002	Al cliente, buscar productos (platos / bebidas) disponibles que se encuentran en el menú a través del nombre, descripción o por precio.
RF003	Al cliente, crear una orden de pedido indicando el número de mesa, la orden de pedido inicia con el estado ABIERTA.
RF004	Al cliente, agregar un ítem de pedido de acuerdo con el producto (plato y bebida) a la orden de pedido, siempre y cuando la orden de pedido se encuentre en estado ABIERTA, POR DESPACHAR (Editar orden de pedido).
RF005	Al cliente, modificar la cantidad del ítem de la orden de pedido, siempre y cuando la orden de pedido se encuentre en estado ABIERTA, POR DESPACHAR (Editar orden de pedido).

RF006	Al cliente, eliminar un ítem de la orden de pedido, siempre y cuando la orden de pedido mientras se encuentre en estado ABIERTA, POR DESPACHAR (Editar orden de pedido).
RF007	Al cliente, visualizar los ítems de la orden de pedido.
RF008	Calcular el precio total de cada ítem de la orden de pedido.
RF009	Calcular el total a pagar de los ítems de la orden de pedido.
RF010	Al cliente, cancelar la orden de pedido, siempre y cuando se encuentre en estado ABIERTA, POR DESPACHAR (Editar orden de pedido).
RF011	Al cliente, registrar su información personal como: nombres, apellidos, tipo de identificación, número de identificación, correo electrónico, número de teléfono, dirección, ciudad y provincia.
RF012	Verificar la existencia de un único cliente a través de su número de identificación.
RF013	Al cliente, visualizar el estado en que se encuentra la orden de pedido.
RF014	Al cliente, registrar los datos personales y dirección al sujeto que desee que se le facture la orden de pedido.

Para el caso de estudio sobre la gestión de boletos de la empresa TRANS&TURIS se aplica el formato de listado, ya que es un formato simple, directo, ágil y fácil de verificar.

Tabla 6 Ejemplo de requisitos funcionales.

Código	**Descripción**
RF01	El sistema permitirá al recaudador de boletería registrar nuevos pasajeros.
RF02	El sistema permitirá al recaudador de boletería realizar reservas de pasajes.

3. Técnicas de Elicitación de Requisitos.

Durante buena parte del texto se ha recalcado la importancia que tienen el usuario durante el proceso de desarrollo, sin embargo, el descubrimiento de los requisitos del sistema se realiza a través de la aplicación de técnicas de recolección de información como la entrevista, la lluvia de ideas, estudios de documentos, análisis de sistemas existentes, entre otras.

En la ingeniería de requisitos se denomina "Elicitación de requisitos" a las actividades de recolectar, reunir y obtener la información. Además, el objetivo principal de la comunicación con los usuarios de la aplicación, es el entendimiento que se debe tener como analistas del problema o sistema en construcción (Washizaki).

Otro aspecto importante en la ingeniería de requisitos, son los procesos de elicitación y análisis que están estrechamente relacionados. Algunos requisitos son descubiertos al aplicar técnicas de elicitación y el resto de la información es analizada por el grupo de analistas para documentar nuevos actores del sistema, problemas que tienen los stakeholders, términos nuevos y requisitos.

Para entender la importancia de aplicar las técnicas de elicitación en la ingeniería de requisitos, se propone un escenario ficticio para obtener información e identificar requisitos. El departamento de boletería de la empresa de transporte TRANS&TURIS será nuestra área de análisis.

3.1. Entrevista.

Al iniciar un nuevo proyecto, la entrevista es una técnica útil, porque los analistas tienen una relación directa con: los expertos del dominio, dueños del producto y/o con cada stakeholder. Además, al ser parte de

las técnicas de elicitación se debe planificar para evitar improvisaciones o inconvenientes al obtener información (González, 2021, p. 52). La entrevista consta de tres fases:

1. Preparación.
2. Conducción / realización.
3. Análisis.

En la fase de preparación, en primer lugar, los analistas deben fijar el objetivo de la entrevista, identificando a la persona que se le aplicará está técnica. En segundo lugar, se debe coordinar la hora y fecha previamente, con el entrevistado para evitar contratiempos. Por último, se debe redactar la lista de preguntas que se aplicará y de ser necesario hacer llegar las preguntas al entrevistado para informarle la temática a tratar.

En la tabla 7, se detalla un ejemplo de preparación de la entrevista, describiendo la fecha, el entrevistador, el entrevistado, objetivo que pretende alcanzar la entrevista y las preguntas que se tratarán.

Tabla 7. Formato de preparación de la entrevista.

Fecha: 20 de junio	Hora: 09h00
Entrevistado: Ing. Javier Gómez	
Empresa: TRANS&TURIS	
Cargo: Gerente	
Entrevistador: Patricia Zumba	
Objetivo: Identificar las necesidades principales de la empresa	
Preguntas: ¿Puede informarme como está marchando el proceso en el departamento de boletería? ¿Y en qué aspectos cree que el sistema o una aplicación podría ayudarle a mejorar esto? ¿Cuántas personas hacen la venta de boletos? ¿Todos ellos venden boletos a cualquier parte? ¿Los empleados de boletería tienen conocimiento o les parecen familiares algún tipo de dispositivo de procesamiento como: ¿computadores, tablets o teléfonos inteligentes?	

En la **fase de conducción** de la entrevista, la experticia del entrevistador es fundamental para establecer un escenario y ambiente ideal con el entrevistado y obtener la mayor cantidad de información. Según (Ambler, 2010), define algunos consejos para mejorar las capacidades al aplicar la entrevista:

1. Envíe con anticipación una agenda acerca de los temas a tratar con el entrevistado, eso le permitirá a él prepararse para ello.
2. Verifique horas antes de la entrevista si está puede aún realizarse, en muchas ocasiones puede resultar incómodo realizarla, debido a que las actividades de la gente suelen cambiar.
3. Agradezca al entrevistado debido a que usted va a tomar algo del tiempo dedicado a sus actividades.
4. Hágale saber a su entrevistado la importancia que él tiene en el sistema o aplicación en desarrollo; indique el proceso en general y por qué la entrevista con él es beneficiosa.
5. No utilice más del tiempo necesario para la entrevista.
6. No interrumpa a su entrevistado cuando habla, muchas ocasiones ello puede desviar la explicación que él intenta dar.
7. No asuma que usted ya sabe lo que va a decir el entrevistado, aun cuando crea haberlo escuchado antes, podría sorprenderse cuantas veces los puntos de vista sobre determinado aspecto no coinciden entre los usuarios.
8. Pregunte lo que más le interese lo más pronto posible, eso permitirá que, si la entrevista debe cancelarse por algún motivo, usted al menos podrá recuperar información importante.
9. Trate de terminar la entrevista resumiendo lo que ha logrado captar de la entrevista, agradezca nuevamente a la persona al finalizar.
10. Trate siempre de identificar las palabras “necesito” y “desearía” generalmente los usuarios quieren obtener cosas maravillosas dentro de sus sistemas, pero en muchas ocasiones no pueden identificar las necesidades mínimas del sistema.

En el siguiente recuadro, se detalla un ejemplo de diálogo entre un experto y la entrevistadora.

El diálogo con el señor Gómez gerente de la empresa TRANS&TURIS.

- **Entrevistadora:** Gracias por recibirme señor Páez, disculpe que tome algo de su tiempo.
- **Sr. Gómez:** No se preocupe, dígame ¿en qué puedo servirle?
- **Entrevistadora:** Estoy a cargo del desarrollo del sistema de venta de boletos y control de reservaciones de la empresa, y, debido a su posición como gerente me interesaría tener su perspectiva acerca de la situación actual, es importante que usted como el encargado de la organización pueda brindarme sus criterios.
- **Sr. Gómez**: Escuchó pues, tiene algo en particular que le interese.
- **Entrevistadora**: ¿Pues sí, puede informarme como está marchando el proceso?
- **Sr. Gómez**: Mire, actualmente el proceso de venta de boletos se ha hecho demasiado tedioso, la cantidad de destinos que cubrimos y los varios turnos que se tienen hacia estos destinos ha hecho que el control manual de las ventas y reservaciones sea mucho más complejo que antes. Desde mi punto de vista esto es un inconveniente para la empresa ya que la imagen que se presenta hacia los usuarios de nuestros servicios se está perdiendo por esta clase de errores.
- **Entrevistadora**: ¿Y en qué aspectos cree que el sistema podría ayudarle a mejorar esto?
- **Sr. Gómez**: Me parece que, a través del uso de ordenadores, las tareas de venta de boletos podrían mejorar la organización y el exceso de papeleo existente en la actualidad, mire que en la actualidad son tres listados que deben actualizarse por la venta de un boleto, esto lleva demasiado tiempo y tienden a cometer muchos errores. Por ejemplo, el listado de pasajeros de un turno para un viaje no siempre se actualiza por errores manuales, vendiendo hasta dos o tres veces el mismo asiento, lo cual causa incomodidad en los pasajeros, he tenido un sinnúmero de quejas por este motivo...

En la **fase de análisis.** En el desarrollo de la entrevista, los analistas van descubriendo algunos detalles con los cuales pueden empezar a plantear los primeros requisitos del usuario. Además, los analistas pueden utilizar otras técnicas de elicitación para analizar y definir los requisitos de la información obtenida de la entrevista. Una de éstas técnicas es la reunión interna (workshop, brainstorm, focus group) entre el grupo de analistas, allí se debaten las ideas y se identifican los requisitos con los que el analista puede empezar son los siguientes:

- El Sistema debería registrar los pasajes vendidos en la lista de pasajeros de cada turno.
- El sistema debe imprimir el boleto entregado a cada pasajero.
- Debe evitar la venta duplicada de asientos en un mismo turno.
- El sistema deberá permitir hacer la reservación de pasaje para un turno específico.
- El sistema deberá controlar la capacidad máxima de cupos de cada bus de la empresa.

Los requisitos que se han detallado en el recuadro anterior, describen la funcionalidad del sistema que deberá hacer una vez que se haya construido. Sin embargo, los no funcionales son importantes porque garantizan la calidad del sistema. Considere el ejemplo siguiente de la empresa TRANS&TURIS:

Otra parte del diálogo con el señor Gómez…

- **Entrevistadora**: … dígame, cuántas personas hacen la venta de boletos.
- **Sr. Gómez**: Actualmente se tienen tres personas en la boletería de las oficinas y dos en las oficinas del terminal terrestre. Las personas que venden los boletos se llaman recaudador de boletería.
- **Entrevistadora**: ¿todos ellos venden boletos a cualquier parte?…

- **Sr. Gómez:** Inicialmente fue así, pero ahora para evitar las confusiones los hemos dividido por destinos, aunque eso no sea tan conveniente, nos gustaría que en cualquier boletería pueda venderse boletos para cualquier destino y que esté disponible día y noche.
- **Entrevistadora:** ¿Ya veo, y los empleados de boletería tienen conocimiento o les parecen familiares los computadores?
- **Sr. Gómez:** No, ninguno de ellos lo tienen, no son expertos en nada de eso

Note, que este fragmento del diálogo se ha podido descubrir nuevas características que resultan ser atributos del sistema, las cuales se detallan a continuación:

Atributos del Sistema

- El sistema debe ser multiusuario (Usabilidad).
- El sistema debe ser de fácil uso (Usabilidad).
- El sistema está disponible 24/7 (Disponibilidad).

3.2. Lluvia de Ideas.

También conocida como brainstorm es una técnica que forma parte de las técnicas grupales como el focus group, workshop, JAD, entre otras. Es útil para la obtención de requisitos, aunque es más compleja para aplicar que la entrevista. Las sesiones suelen estar formadas por un grupo de cuatro a diez participantes. Uno de los cuales hace de jefe de la sesión (Piattini Velthuis, 1996, p.54).

La lluvia de ideas pretende que su discusión puede de cierta manera poner en evidencia aquellos requisitos reales basados en las ideas de todos los usuarios del sistema o del grupo de analistas. Dependerá donde se aplique, por ejemplo:

- Analizar la información obtenida de la entrevista entre los integrantes del grupo de análisis para formular de forma correcta los requisitos.
- Descubrir otros requisitos que no están claros incluso si se han aplicado otras técnicas de elicitación.
- Para integrar a otros usuarios del sistema.
- Para la gestión de requisitos al generar ideas relacionadas con los requisitos del producto o servicio.

En la figura 8, se detallan las actividades que se llevan a cabo en una lluvia de ideas convencional utilizando material de oficina, y guiado por facilitadores. En la primera fase, el facilitador inicia la motivación y plantea las ideas generales para que los stakeholder generen un conjunto de ideas, luego, se ordena las ideas y se llega un consenso de las ideas más brillantes.

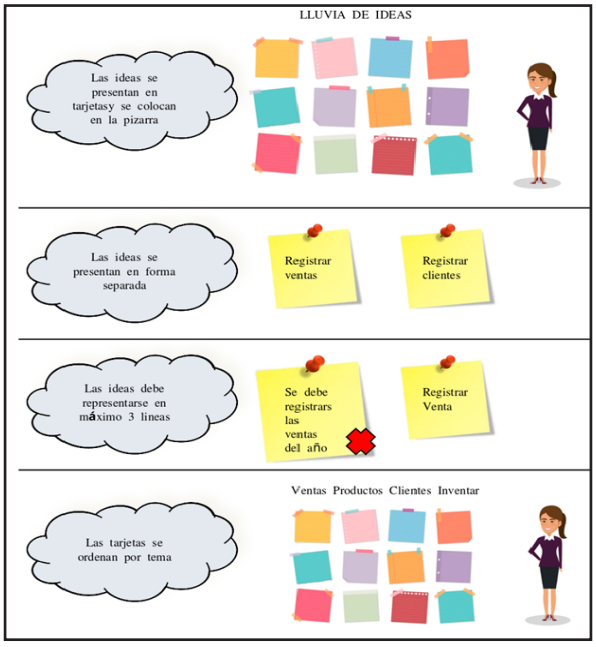


Figura 8. Proceso de lluvia de ideas. Tomado de (Lluvia De Ideas. Técnicas Participativas, 2012).

La intención de aplicar esta técnica es generar la mayor cantidad de ideas, luego se irán eliminando dependiendo de su complejidad, originalidad, aplicabilidad. Se definen algunas reglas básicas:

1. Los participantes deben tener mucha experiencia en los temas a discutir y pertenecer a distintas disciplinas.
2. Conviene suspender el juicio crítico y permitir la evolución de cada idea, sino se creará un ambiente hostil.
3. No se debe descartar ninguna idea por más desequilibrada o excéntrica que parezca, luego se las plantea y pueden convertirse en requisitos potenciales del sistema.
4. A veces una idea resulta en otra idea, y otras veces podemos relacionar varias ideas para generar una nueva.
5. Escribir todas las ideas sin censura.

La lluvia de ideas al igual que otras técnicas se debe planificar para evitar improvisaciones o inconvenientes al obtener la información (Piattini Velthuis, 1996). Se deben seguir cuatro fases para desarrollar de forma adecuada esta técnica:

1. Preparación.
2. Generación.
3. Consolidación.
4. Documentación.

En la **fase de preparación**, en primer lugar, el equipo de analista debe establecer el objetivo que pretende alcanzar la lluvia de ideas. Otro elemento indispensable es seleccionar a las personas que integrarán el panel quienes generarán las nuevas ideas. En segundo lugar, la elección del rol moderador, quién se encargará de manejar el grupo y tendrá que explicar al inicio cuales son las reglas y los temas a discutir. Un punto importante es coordinar previamente la hora y fecha que se llevará

a cabo la reunión para evitar contratiempos. Para finalizar, se debe redactar los temas o preguntas que serán discutidas en la reunión.

En el caso de estudio de TRANS&TURIS, se ha planificado el brainstorm a los asistentes de boletería a través de dos preguntas detalladas en la tabla 8.

Tabla 8. Formato de preparación de la lluvia de ideas.

Fecha: 25 de junio	**Hora**: 17h00
Moderador: Ing. Wilman Chamba	
Empresa: TRANS&TURIS	
Personas participantes: - María González. - Darío Villata. - Alexandra Sarango. - Emilio Ortega.	
Objetivo: Discutir ideas para la implementación del sistema de boletería.	
Preguntas para el debate: ¿Cuáles son las tareas en la venta de boletos que se pueden automatizar? ¿Qué errores se deben evitar en la venta de boletos? ...	

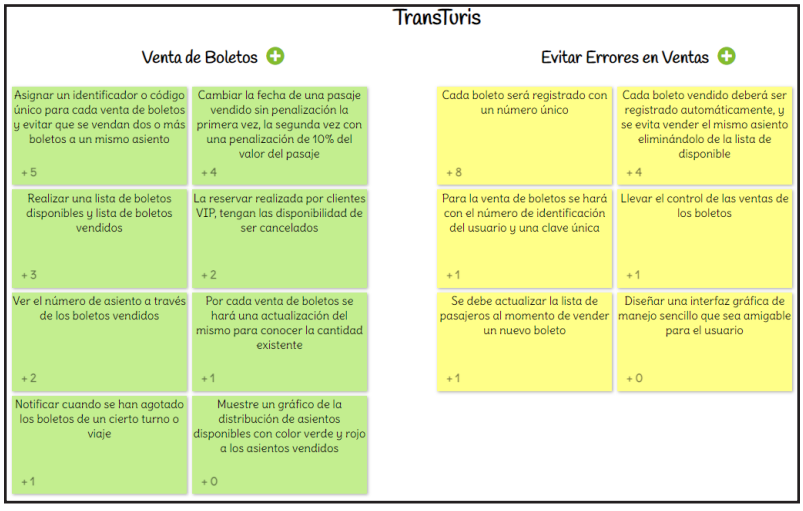


Figura 9. Aplicación de la técnica de lluvia de ideas a través de ideaboardz.

En la **fase de generación, consolidación y documentación** el grupo de desarrollo puede aplicar la técnica de forma convencional a través de una pizarra, stickers, etc. o apoyados por herramientas web como ideaboardz[2] o miro[3].

En primer lugar, el moderador plantea la pregunta o el tema a tratar "¿Cuáles son las tareas en la venta de boletos que se pueden automatizar?" y debe motivar la cooperación de los participantes. En segundo lugar, los participantes analizan y desarrollan las ideas, las anotan y exponen. Finalmente, el moderador pone a discusión todas las ideas y se solicita votar por las ideas más sobresalientes que sean factibles de implementar. En la figura 9 se ilustra la pizarra de la herramienta ideaboardz donde los usuarios listan sus ideas que son sujetas a valoraciones por parte de todos los participantes.

En las tablas 9 y 10, muestra el reporte de ideas generado por la herramienta *ideaboardz* ordenado en forma descendente con respecto a su valoración.

Tabla 9. Ideas para el sistema de boletería - Venta de boletos.

Venta de boletos	Votos
Asignar un identificador o código único para cada venta de boletos y evitar que se venden dos o más boletos a un mismo asiento.	5
Realizar una lista de boletos disponibles y lista de boletos vendidos.	3
Los boletos vendidos tengan la disponibilidad de ser cancelados.	2
Ver el número de asiento a través de los boletos vendidos.	2
Por cada venta de boletos se hará una actualización del mismo para conocer la cantidad existente.	1
Notificar cuando se han agotado los boletos de un cierto turno o viaje.	1
Mostrar un gráfico de la distribución de asientos, en donde los asientos pintados de color verde represente que están disponibles y los de color rojo que están vendidos.	0

2 https://ideaboardz.com/
3 https://miro.com/

Tabla 10. Ideas para el sistema de boletería - Evitar errores en ventas.

Evitar errores en ventas	Votos
Cada boleto será registrado con un número único.	8
Cada boleto vendido deberá ser registrado automáticamente, y se evita vender el mismo asiento eliminándolo de la lista de disponible.	4
Para la venta de boletos se hará con el número de identificación del usuario y una clave única.	1
Llevar el control de las ventas de los boletos.	1
Se debe actualizar la lista de pasajeros al momento de vender un nuevo boleto.	1
Diseñar una interfaz gráfica de manejo sencillo que sea amigable para el usuario.	0

Las ideas con mayor valoración se toman como referencia para convertirlas en requisitos del usuario. Los requisitos obtenidos aplicando la lluvia de ideas son:

- El sistema asignará un identificador único a cada boleto vendido asociado a un asiento único para evitar inconsistencia.
- El sistema visualizará la lista de pasajes vendidos.
- El sistema visualizará la lista de asientos disponibles.
- El sistema permitirá al recaudador de boletería cancelar una reserva a clientes vip.
- El sistema permitirá al recaudador de boletería cambiar la fecha del pasaje por una ocasión sin penalización, la segunda vez tendrá una sobrecarga del 10 % del pasaje.
- El sistema realizará una actualización a la lista de pasajeros por cada boleto vendido.
- El sistema validará la disponibilidad de pasajes para un determinado turno, mostrando una notificación indicando que "no existen pasajes disponibles para el determinado turno".

3.3. Estudios de documentos.

La mayoría del conocimiento sobre el dominio del problema viene de la lectura y/o revisión de documentos. El análisis o estudio de documentos forma parte de las técnicas cognitivas útiles para identificar requisitos. Es una técnica que consiste en realizar la lectura o revisión de documentos referentes al dominio del problema sobre la información que fluye dentro de la empresa que pueden ser: reglamentos, contratos, memorandos, manuales de técnicos, comprobantes de pago, recibos, boletas, guías de capacitación, entre otros (Wong, 2017, p. 55), o fuera de ella como leyes, normativas, disposiciones.

Es una técnica útil para encontrar un conocimiento profundo sobre una tarea en particular. Lo importante de esta técnica es que no se necesita de la intervención del experto salvo que se necesite de su autorización para acceder a los documentos, sino, se puede realizar de forma autónoma (Gottesdiener, 2005, p. 91). El objetivo es recolectar la información necesaria para el proceso de documentación de requisitos. Además, esta técnica sirve más como complemento de las otras técnicas explicadas, y se debe plantear las siguientes preguntas:

1. ¿Cuál es el propósito de este documento?
2. ¿Quién lo usa? ¿Por qué? ¿Para qué?
3. ¿Cuáles son las tareas que realizan con este documento?
4. ¿Se puede encontrar una relación entre los documentos?

El estudio de documentos debe planificarse a través de tres fases:

1. Preparación.
2. Aplicación.
3. Consolidación.

En la **fase de preparación**, los analistas seleccionan el material a estudiar y se plantean las preguntas que deben ser contestadas en la lectura o análisis del documento. En la tabla 8 se detalla el material seleccionado "**REGLAMENTO A LEY DE TRANSPORTE TERRESTRE TRÁNSITO Y SEGURIDAD VIAL**", artículo 46 y las tres preguntas planteadas para el caso de estudio TRANS&TURIS.

Tabla 11. Formato de preparación del estudio de documentos.

Fecha: 28 de junio	**Hora: 13h00**
Plataforma en la que fue encontrada: Enlace	
País: Ecuador	
Año de publicación: 2012	
Preguntas: ¿Cuáles son los porcentajes de descuentos en los pasajes? ¿Quiénes se benefician de los descuentos? ¿Qué tipos de tarifa existen? …	

En la **fase de aplicación**, el equipo de analistas estudia la documentación, subraya las partes importantes y da contestación a las preguntas formuladas.

Art. 46.- Tendrán derecho a las tarifas preferenciales:

1. Las personas con discapacidad que cuenten con el carné o registro del Consejo Nacional de Discapacidades, según el artículo 20 de la Ley sobre Discapacidades, pagarán una tarifa preferencial del 50 % en el transporte terrestre, y el servicio prestado será en las mismas condiciones que los demás pasajeros que pagan tarifa completa.
2. **Los estudiantes de los niveles básico y bachillerato** que acrediten su condición mediante presentación del carné estudiantil otorgado por el Ministerio de Educación, pagarán una tarifa preferencial del 50 % bajo las siguientes condiciones:

3. **Los estudiantes de los niveles básico y bachillerato** que acrediten su condición mediante presentación del carné estudiantil otorgado por el Ministerio de Educación, pagarán una tarifa preferencial del 50 % bajo las siguientes condiciones:
   a) Que el servicio lo utilicen durante el periodo o duración del año escolar.
   b) Que lo utilicen de lunes a viernes.
   c) Los días sábados, por situaciones especiales como desfiles cívicos, participaciones comunitarias, eventos académicos, culturales y deportivos estudiantiles, pagarán una tarifa preferencial del 50 % en el transporte terrestre.
4. **Las niñas, niños y adolescentes**, pagarán una tarifa del 50 %. Los niños, niñas y adolescentes hasta los 16 años de edad no estarán en la obligación de presentar ningún documento que acredite su edad. Los adolescentes estudiantes desde los 16 años de edad en adelante accederán a la tarifa preferencial mediante la presentación de su cédula de identidad.
5. **Las personas mayores de 65 años** que acrediten su condición mediante la presentación de la cédula de ciudadanía o documento que lo habilite como tal, pagarán una tarifa preferencial del 50 % en todo el transporte terrestre.

En todos los casos, el servicio prestado será en las mismas condiciones que los demás pasajeros que pagan tarifa completa.

Finalmente, en la **fase de consolidación,** la documentación analizada proporciona información fundamental para: la identificación de términos desconocidos (tarifa preferencial) para el glosario de términos, nuevos actores del sistema (estudiantes, adultos mayores, menores de edad, personas con discapacidad) y para los requisitos del sistema. Este tipo de información debe ser documentada con el objetivo de robustecer el documento de especificación de requisitos.

Glosario de términos:

- Tarifa preferencial: es el descuento del 50 % al pasaje total que se aplica a las personas con discapacidad, estudiantes de nivel básico y bachillerato; niñas, niños y adolescentes; y personas mayores de 65 años.

Actores:

- Personas con discapacidad.
- Estudiante de nivel básico y bachillerato.
- Niñas, niños y adolescentes.
- Personas mayores de 65 años.

Requisitos:

- El sistema deberá clasificar a un nuevo cliente con tarifa preferencial (personas con discapacidad; estudiante de nivel básico y bachillerato; niñas, niños y adolescentes; y personas mayores de 65 años).
- El sistema deberá descontar el valor del pasaje a los pasajeros con tarifa preferencial.

3.4. Estudio de sistemas existentes.

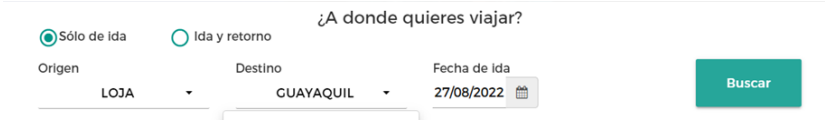
Otra técnica útil es, el estudio de distintos sistemas existentes o que se encuentran operando relacionados con el sistema o aplicación a ser construido. El objetivo es analizar en primera instancia las interfaces gráficas del usuario o funcionalidad, observando el tipo de información que se maneja y los procesos que se realizan. En otras palabras, el analista identifica entradas, procesos y salidas. Además, permite agregar nuevas funcionalidades no adoptadas en otras técnicas de elicitación.

Lo importante de esta técnica es que no se necesita de la intervención del experto, sino que se puede realizar de forma autónoma, para ello en la web existe un sinnúmero de sistemas para analizar, por ejemplo, en el caso de estudio TRANS&TURIS se toma como referencia de estudio la aplicación ISYPLUS Pasaje, con respecto a la funcionalidad sobre la búsqueda de un

pasaje de bus en la ruta Loja a Guayaquil, el objetivo principal es observar que tipo de operaciones se realiza y la información presentada. En las tablas 12, 13 y 14 se observa cómo se puede esquematizar la información del análisis de estudio de un sistema existente.

Tabla 12. Búsqueda de pasajes en el sistema ISYPLUS Pasaje.

Fecha: 24 de agosto.	**Hora:** 10h00.
Tema: Búsqueda de pasajes.	
Plataforma que fue encontrada: https://www.isyplus.com/v2/pasajes#/main/	



Entradas:

- Origen (lista ciudad).
- Destino (lista de ciudad).
- Fecha del pasaje.

Procesos:

- El usuario ingresa los criterios: origen de la ciudad, destino de la ciudad y la fecha, para buscar rutas.
- El sistema busca las rutas con cada uno de sus turnos de acuerdo a los criterios seleccionados por el usuario.

Tabla 13. Lista de pasajes del sistema ISYPLUS Pasaje.

Fecha: 24 de agosto.	**Hora:** 10h00.
Tema: Búsqueda de pasajes.	
Plataforma que fue encontrada: https://www.isyplus.com/v2/pasajes#/main/	

Entradas:

- Selección de turno.
- Selección de asiento.

Salidas:

- Lista de turnos (hora salida, ciudad de origen, ciudad de destino, y el precio).

Procesos

- El sistema muestra la lista de turnos disponible.
- El usuario selecciona el asiento en un turno específico.

Tabla 14. Selección de asientos del sistema ISYPLUS Pasaje.

Fecha: 24 de agosto. **Hora:** 10h00.

Tema: Búsqueda de pasajes.

Plataforma que fue encontrada: https://www.isyplus.com/v2/pasajes#/main/

Entradas:

- Número de asiento.
- Cédula.
- Nombres y apellidos.

Salidas:

- Precio.
- Fecha.
- Hora.
- Ciudad de Origen.
- Ciudad de Destino..

Proceso:

- El sistema muestra la distribución de asiento que tiene el turno, asiento de color:
  - Rojo, ocupados.
  - Blanco, disponible.
  - Verde, seleccionados.
- El usuario selecciona un turno disponible.
- El sistema muestra una tabla para el ingreso de los datos del pasajero.

3.5. Prototipos.

En la recolección de requisitos, hay la posibilidad de que algunos no estén demasiados claros y que el grupo de analistas tenga dudas. Entonces, el prototipo es una técnica que tiene por objetivo clarificar requisitos ambiguos a través de representaciones de interfaces gráficas para el usuario. Los prototipos garantizan la calidad puesto que permiten conseguir una importante retroalimentación a requisitos previamente identificados. Además, los prototipos son una versión inicial del sistema y son usados para ayudar a elicitar y validar requisitos.

Existe un amplio rango de técnicas de prototipos para requisitos:

- Diseños de plantillas dibujadas en papel llamado prototipos de papel.
- Versiones betas del producto de software llamado wireframes.
- Versiones profesionales de las interfaces gráficas de software llamado mockup.

El **prototipo de papel** es una técnica que se basa en la utilización de materiales básicos como lápiz, borrador, tijeras o una hoja para crear pequeñas interfaces de usuarios en cortos periodos de tiempo. El objetivo es verificar si los stakeholders pueden realizar las tareas usando

estas interfaces y evaluar que se cumplan con los requisitos planteados (NIELSEN, 2014).

El **prototipo basados en wireframes** se base en el diseño de interfaces gráficas que contendrá el sistema utilizando herramientas gráficas como pencil[4] o ninjamock[5]. Wireframe es una mejora al prototipo de papel porque proponen la estructura o esqueleto de todas las interfaces del sistema priorizando la funcionalidad, resultados, disposición de los contenidos, pero evitando el diseño profesional basado en colores o estética (Warfel, 2009, p. 19).

Por último, el **mockup** es una técnica que realiza representaciones completamente estáticas del diseño visual. Es una mejora al prototipo basado en wireframes, que detalla el diseño final de las interfaces gráficas del usuario. Esta técnica se utiliza en la fase de diseño en las metodologías de desarrollo de software. Además, se lo conoce como prototipo de diseño profesional (Mannio, 2010).

Los prototipos son indispensables en el proceso de validación de requisitos, al igual que todas las técnicas de elicitación deben planificarse. Según (Silva et al., 2020), los prototipos se aplican en diferentes reuniones con los stakeholder, por ejemplo:

1. En la primera reunión, el equipo de analistas debe tomar notas de los comentarios de los stakeholders y dibujar bocetos rápidos en un papel.
2. Finalizada la reunión, el equipo elabora un prototipo más robusto en papel o con una herramienta gráfica (Pencil por ejemplo).
3. En una segunda reunión, los miembros del equipo validan este prototipo con el stakeholder. Este prototipo también se utiliza para

4 https://pencil.evolus.vn/
5 https://ninjamock.com/

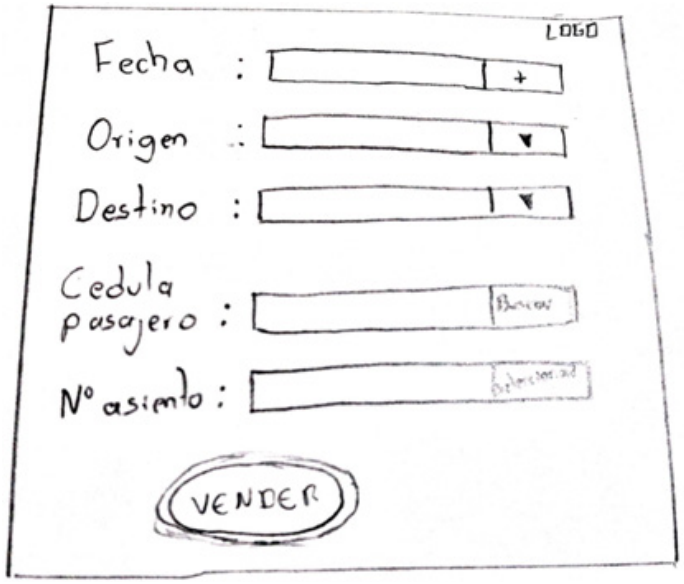
extraer nuevos requisitos o para especificar otros requisitos que no estaban tan claros en la primera reunión. Durante la reunión, el equipo puede hacer correcciones en el prototipo.

4. Después de la segunda reunión, el equipo de analista reconstruye el prototipo en detalle con todas las correcciones y ampliaciones extraídas de la segunda reunión en un papel o con una herramienta gráfica.
5. Si es necesario, en la tercera o cuarta reunión los miembros del equipo vuelven a validar los nuevos prototipos. En estas últimas reuniones el cliente revisa los prototipos y propone las últimas correcciones a las inconsistencias.
6. Al final, con las últimas correcciones sobre los prototipos, el equipo elabora el documento de especificación de requisitos.

Para el caso de estudio del sistema TRANS&TURIS se aplicó las técnicas del prototipo basado en papel y wireframe. Se diseñaron dos prototipos de papel para el proceso "Buscar pasaje". El objetivo es representar los requisitos a través de una interfaz gráfica. En reunión convocada para asignar esta actividad, se solicitó a dos analistas que realicen el prototipo con la información recopilada.

En la figura de la tabla 15 se detalla la interfaz para buscar pasaje(s) en una ruta o turno, se observa que este prototipo incluye correctamente la ciudad de origen, ciudad de destino y la fecha de viaje para realizar la búsqueda de los turnos según los datos ingresados. Además, en esta pantalla se observa que existe una saturación de datos que no son necesarios y por ende no están relacionados con la búsqueda de los pasajes disponibles, como son, el número de identificación del pasajero (Cédula Pasajero) y número de asiento; esa información debería ser solicitada en otro proceso y pantalla como la venta del boleto.

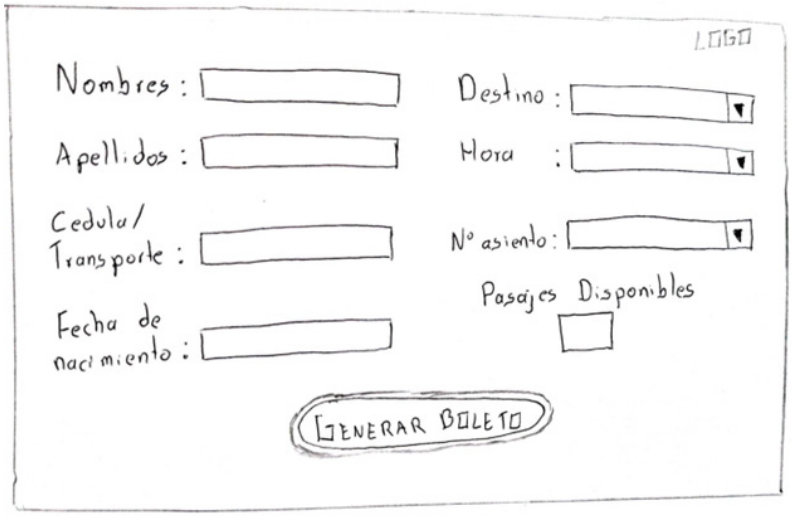
Tabla 15. Buscar pasaje(s) de la aplicación TRANS&TURIS (Analista 1).

Pantalla propuesta para Buscar Pasaje(s) de la aplicación TRANS&TURIS		
Nro. Pantalla:	001.	Versión: 0.1.
ROL: Analista 1.		
Observaciones:	• Para la búsqueda de pasajes se necesita de la fecha de viaje, del origen y destino de la ciudad. • Existe saturación de datos como la cédula del pasajero y el número de asiento. • El botón vender debe cambiarse por “buscar ruta”.	
		

En la figura de la tabla 16 se presenta otra propuesta de prototipo para buscar pasajes en una ruta o turno y se solicita al usuario interesado ingresar datos informativos como: nombres, apellidos, cédula, fecha de nacimiento, número de asiento. Sin embargo, las entradas indicadas hacen referencia a registrar un pasajero y no tiene relación con el proceso de buscar pasajes. Además, se omite una entrada importante como es la ciudad de origen para buscar pasajes en una ruta o turno.

Tabla 16. Buscar pasaje(s) de la aplicación TRANS&TURIS (Analista 2).

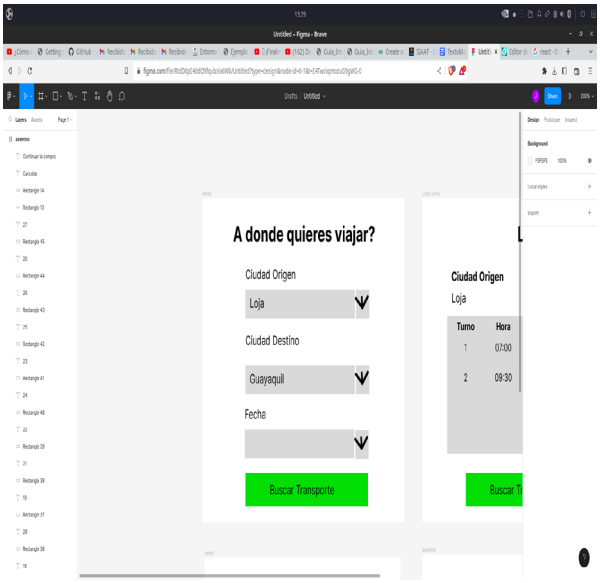
Pantalla propuesta para Buscar Pasaje(s) de la aplicación TRANS&TURIS		
Nro. Pantalla:	001.	V 0.2.
ROL: Analista 2.		
Observaciones:	• Para la búsqueda de pasajes se necesita de la fecha de viaje, ciudad de origen y la ciudad destino. • Las entradas de nombres, apellidos, cédula, fecha de nacimiento, hora, número de asiento y pasajes disponibles deben ser omitidas debido a que no son importantes para buscar un pasaje. • Agregar las entradas "destino" y fecha. • El botón "Generar Boleto" se debe cambiar a "Buscar Pasaje".	



El prototipo de papel cumple la función de verificar las actividades y acciones que tienen que realizar los usuarios del sistema con la interfaz (pantallas), por lo tanto, estas deben estar claras y entendibles para los usuarios, de lo contrario, el analista debe realizar los ajustes necesarios en el prototipo con ayuda de los stakeholder.

Después de corregir los prototipos de papel y hayan sido validados por los usuarios, se diseñan en herramientas tipo Wireframe para organizar la disposición de los elementos de la arquitectura de la información. Para el caso TRANS&TURIS se diseñó los prototipos en la herramienta pencil, como se detalla en las tablas 17 a 20.

Tabla 17. Prototipo final para "Buscar turnos" de la aplicación TRANS&TURIS.

Pantalla Buscar Turnos de la aplicación TRANS&TURIS		
Nro. Pantalla:	001.	V 0.3.
ROL: Analista 1.		
Observaciones:	• En la búsqueda de turnos son necesarios los criterios de: ciudad de origen, ciudad destino y la fecha de búsqueda. • Busca los turnos disponibles de acuerdo a los criterios de búsqueda ciudad de origen, destino y fecha. • Lista los turnos disponibles de acuerdo a los criterios de búsqueda. • Para este tipo de operaciones no es necesario que el usuario esté autenticado.	
		

En las siguientes tablas se muestra únicamente la versión final de los prototipos 002, 003 y 004 desarrollados en la herramienta pencil.

Tabla 18. Lista de turnos de la aplicación TRANS&TURIS.

Pantalla Lista de turnos de la aplicación TRANS&TURIS	
Nro. Pantalla:	002. V 0.3.
ROL: Analista 1.	
Observaciones:	Se detalla la lista de turnos disponibles.

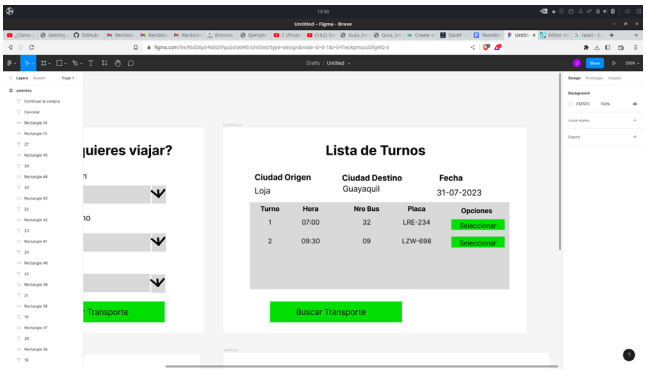


Tabla 19. Ingreso a la aplicación TRANS&TURIS.

Pantalla Ingreso al Sistema de la aplicación TRANS&TURIS	
Nro. Pantalla:	003. V 0.3.
ROL: Analista 1.	
Observaciones:	

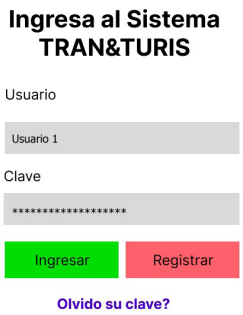
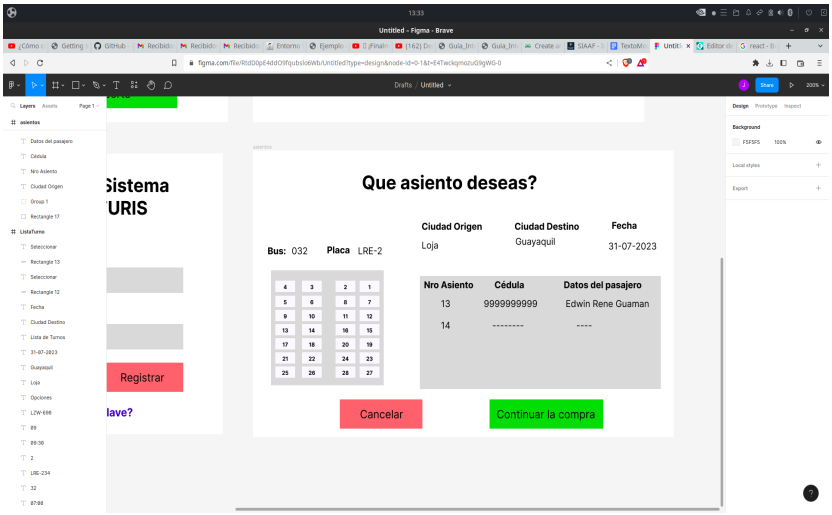


Tabla 20. Selección de asientos de la aplicación TRANS&TURIS.

Pantalla Seleccionar Asiento de la aplicación TRANS&TURIS		
Nro. Pantalla:	004.	V 0.4.
ROL: Analista 1.		
Observaciones:		



Asimismo, en esta etapa se puede describir la interacción que existe entre las diferentes interfaces gráficas, esta interacción es estática, donde se puede ir detallando a nivel del wireframes la navegabilidad entre diferentes botones, enlaces u opciones, mostrando dónde nos va a llevar la acción. En el mismo contexto, se aconseja agregar notas para explicar que sucede al presionar cualquier componente que exija navegabilidad. A continuación, se indica la navegación que existe entre las diferentes pantallas en una herramienta de prototipado, en este caso se utilizó Pencil.

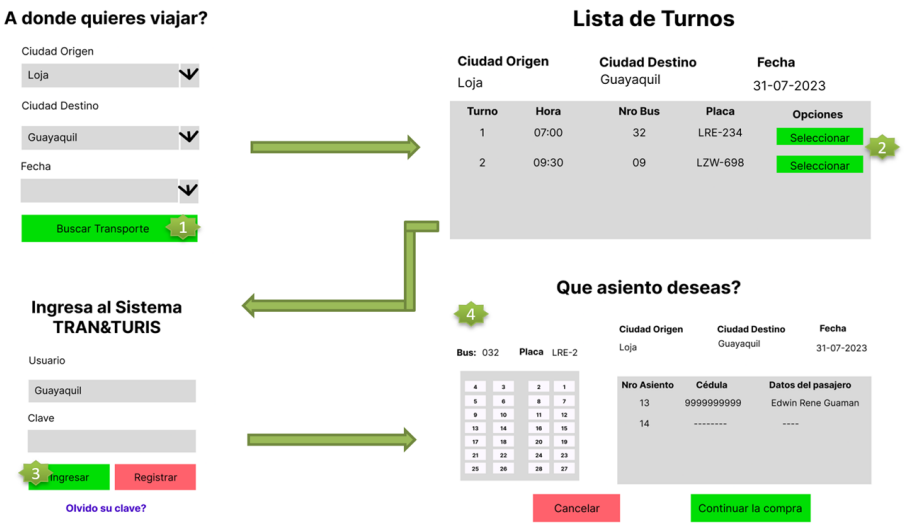


Figura 10. Ejemplo de navegabilidad de prototipo.

Los prototipos han permitido a los analistas ir descubriendo requisitos ocultos o aclarar requisitos que posean ambigüedades. En la siguiente tabla se detallan algunos requisitos descubiertos que se omitieron inicialmente:

- El sistema permitirá cargar las ciudades de destinos dependiendo de la ciudad de origen seleccionada.
- El sistema permitirá realizar búsqueda de rutas de acuerdo a los criterios de búsqueda seleccionado (ciudad de origen, ciudad destino, fecha).
- El sistema mostrará la lista de turnos disponibles (turno, hora, placa y número de bus).
- El sistema permitirá al usuario seleccionar un turno disponible.
- El sistema permitirá mostrar la distribución de asientos del turno seleccionado.
- El sistema permite elegir un asiento disponible.

3.6. Construcción del glosario de términos.

En una gran cantidad de ocasiones, la terminología o jerga usada en ciertos dominios tiene diversos significados, aún más, existen ciertos términos o palabras que existen solamente como parte de la terminología de un dominio. El glosario de términos permite dar definiciones a los sustantivos o frases sustantivas que forman parte del dominio del problema y que, aunque las conozcamos permite que todos quienes intervienen en el desarrollo de un sistema unifiquen sus criterios en cuanto a su significado.

Técnicamente un glosario de términos es un documento que define todos los términos que forman parte del dominio del problema, con el objetivo que el equipo de desarrollo y personas involucradas (stakeholders) manejen el mismo lenguaje y no existan varias interpretaciones. Por ejemplo, en el dominio del problema que está en estudio, considere las siguientes definiciones.

Tabla 21. Glosario de términos de la aplicación TRANS&TURIS.

Término	Descripción / Definición
Bus.	Vehículo terrestre para transporte masivo.
Pasaje.	Comprobante del pago entregado al usuario para que lo presente al controlador durante el viaje.
Tarifa preferencial.	Es el descuento del 50 % al pasaje total que se aplica a las personas con discapacidad, estudiantes de nivel básico y bachillerato; niñas, niños y adolescentes; y personas mayores de 65 años.
Viaje.	Trasladarse de un lugar a otro.
Turno.	La fecha y hora programada de un viaje.
Ruta.	Camino establecido o previsto para un viaje.
Asiento.	Lugar donde se ubicará el pasajero dentro del vehículo de transporte.

A veces, el desarrollo de los sistemas falla debido a las múltiples interpretaciones que se hace acerca de los elementos que intervienen en el dominio del problema o del desconocimiento que se tiene de ellos. Por eso es útil, definir el glosario para que todas las personas involucradas en el desarrollo entiendan lo que el resto están entendiendo sobre cada elemento del dominio.

4. Modelo del Dominio.

En el proceso de determinación de requisitos, los analistas se habrán empapado de un conjunto de términos nuevos o conocidos que tienen un significado diferente en el dominio del problema en estudio. Cada uno de ellos tendrá una particular importancia durante la ejecución de la actividad relacionada con el modelo del dominio.

Por este motivo, el modelo del dominio es un modelo del sistema en el que se representan gráficamente conceptos del mundo real y sus relaciones que intervienen en el dominio del problema o en sus reglas de negocio. El proceso de modelamiento del dominio consiste en encontrar esos objetos y sus relaciones tal y como intervienen en la realidad. Además, al modelo del dominio inicial también se lo conoce como modelo conceptual.

Es importante acotar, que conceptos de implementación o soluciones técnicas del problema no se los debe modelar o hacer constar en el diagrama, como por ejemplo los términos: sistema, base de datos, interfaz, ventana, archivos, impresora, etc.

4.1. Concepto, Abstracción y Definición.

4.1.1. Concepto

Un concepto es una representación mental que hacemos con relación a los objetos que existen en el mundo real. Así, podemos decir que, cada idea que nuestra mente almacena con relación a un objeto, es un concepto.

Por lo general los humanos asociamos a cada concepto una palabra diferente del lenguaje, aunque debe considerarse el hecho de que una misma palabra pueda estar asociada a más de un concepto. Cada

persona maneja un conjunto de conceptos de acuerdo a un dominio específico. Durante el proceso de desarrollo, es muy conveniente que el analista coincida en conceptos con aquellos manejados por el experto del dominio, debido a que ello evitará malentendidos en un futuro.

Existen objetos que tienen existencia física en el mundo real, por ejemplo, un bus, un boleto, asiento, un comprobante como factura, etc.; pero también hay conceptos que existen pero que no se los puede visualizar, es decir, carece de presencia física, como por ejemplo, una cuenta, penalización, entre otras; a este tipo de conceptos se los denomina conceptos abstractos.

4.1.2. Abstracción

El proceso a través del cual cada persona captura y almacena conceptos en su mente es conocido como abstracción. Abstraer significa tomar los detalles más significativos de un objeto y formar una idea o concepto con esos detalles.

La abstracción es un proceso que puede lograrse a través de mecanismos como la observación o a través de la transmisión de conocimientos. Puede entonces que, en parte, la aplicación de las técnicas de elicitación como la entrevista, la observación, lluvia de ideas, etc., tratan de hacer que el analista encuentre las abstracciones necesarias del dominio del problema, a través de la transmisión de conocimiento del experto en el área.

Además, el objetivo no es que el analista se convierta en un experto sobre el dominio, pero si hay que decir que, si el analista no comprende adecuadamente un problema, un dominio o las reglas del negocio, es bastante probable que no pueda darle una solución adecuada.

4.1.3. Definición

La definición es la descripción que se hace acerca de un concepto particular. En esencia puede entenderse a la definición como la descripción que se hace acerca de la naturaleza de un objeto, es decir, la respuesta a la pregunta ¿qué es?.

4.2. Descubriendo los conceptos del dominio.

El objetivo de la construcción del modelo conceptual es el de encontrar los conceptos del dominio y las relaciones que existen entre ellos.

El primer paso es encontrar los conceptos propios del dominio del problema, se utilizará el enfoque orientado a objetos como se detalla en la figura 11.

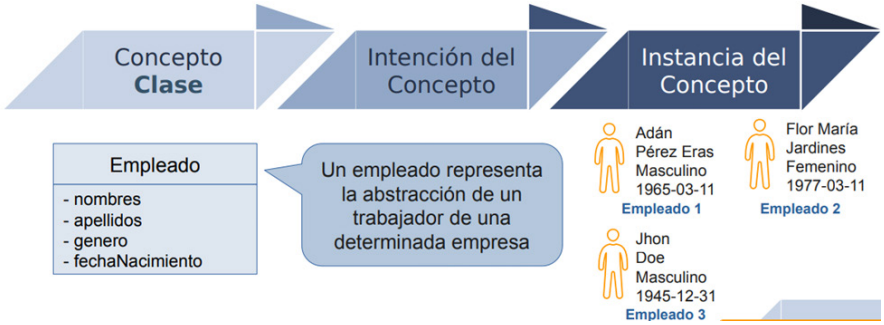


Figura 11. Identificando los conceptos del problema.

Durante la fase de análisis del problema, los analistas adquieren toda la información necesaria acerca del dominio del problema. Mucho tiempo se destina a las entrevistas, lluvias de ideas, observación del negocio, etc. La cantidad de nueva información que el analista ha adquirido es importante, pues si se considera que se está hablando con un experto, es mucho lo que él puede transmitirle con respecto al negocio.

La identificación de los conceptos es una de las tareas complejas de la etapa de análisis y en particular de la construcción del modelo del dominio, sin embargo, es la experiencia y el tratar día a día de construir modelos mentales del mundo lo que más ayuda en este proceso.

En la figura 12 se detalla el proceso de construcción del modelo del dominio que en primera instancia es la de descubrir conceptos relacionados al problema a través de la inspección gramatical. El segundo paso es la depuración de la lista de conceptos duplicados, mal escritos, etc. El tercer paso, es dibujar a través de un diagrama de clases de UML (*Unified Model Language*, Lenguaje Unificado para el Modelado), los conceptos identificados con sus atributos, relaciones y multiplicidad si es posible.

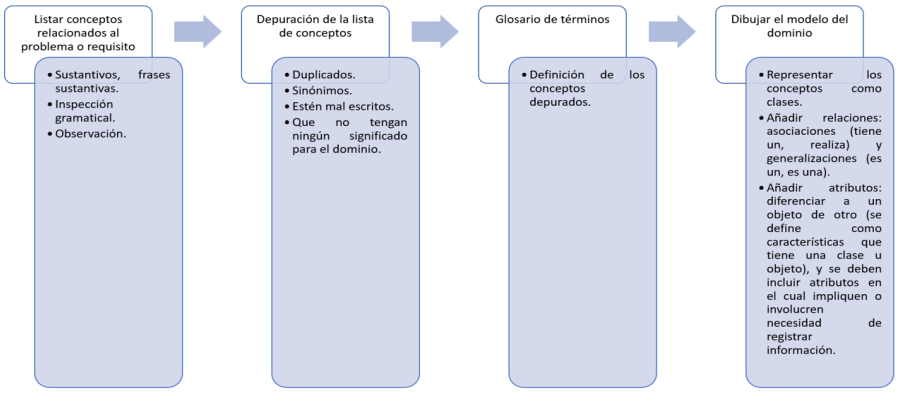


Figura 12. Guía para el modelado del dominio.

4.2.1. Técnica de Inspección gramatical.

La técnica de inspección gramatical usa la forma más simple de encontrar los conceptos del dominio, realiza una exploración de los documentos redactados durante la fase de determinación de requisitos en busca de sustantivos o frases sustantivas. Por ejemplo, en el detalle

del problema de TRANS&TURIS es común encontrar términos como boleto, ruta, pasaje, turno, pasajero.

La técnica tiene muchas ventajas. En primer lugar, los lenguajes naturales o humanos son comprendidos por todos quienes participan en un proyecto de desarrollo y pueden ser un medio adecuado para la comunicación entre expertos y analistas.

Además, la técnica no requiere de aprendizajes adicionales y puede ser aplicada únicamente sobre los documentos existentes en la etapa de determinación de requisitos. A continuación, se detallan los pasos necesarios para aplicar la técnica de inspección gramatical:

1. Lea cada uno de los documentos obtenidos en la etapa anterior.
2. Durante la lectura subraye o resalte cada uno de los sustantivos o frases sustantivas encontrados.
3. Saque los elementos subrayados en una pieza de papel separada.

Es importante que, al tiempo que vaya ejecutando la última actividad, actualice el glosario de términos cuando encuentre un concepto relevante. Esto le dará información con la cual se puede depurar luego la lista de conceptos encontrada.

Se tiene una gran posibilidad de encontrar la gran mayoría de conceptos del dominio. Esto es conveniente, pero depende de la capacidad del analista para encontrar y redactar adecuadamente los artefactos de documentación de la etapa de requisitos.

La inspección gramatical con analistas experimentados es un proceso mental de cada una de las fases para seleccionar las clases candidatas. Sin embargo, para analistas sin experiencia es recomendable realizar cada paso para entender el proceso de inspección gramatical.

Por ejemplo, la inspección gramatical realizada sobre los requisitos funcionales iniciales de la empresa TRANS&TURIS es la siguiente:

La **SISTEMA** deberá permitir:

- Al **RECAUDADOR DE BOLETERÍA** administrar **DESTINOS** y **TURNOS.**
- Al recaudador de boletería registrar nuevos **PASAJEROS.**
- Registrar los **PASAJES** vendidos en la **LISTA DE PASAJEROS** de cada **TURNO.**
- Imprimir o descargar el **BOLETO** entregado a cada **PASAJERO.**
- Evitar la **VENTA** duplicada de **ASIENTOS** en un mismo turno.
- Realizar la **RESERVACIÓN** de pasaje para un turno específico.
- Controlar la **CAPACIDAD MÁXIMA** de **CUPOS** de cada **BUS** de la **EMPRESA.**
- Asignar un **IDENTIFICADOR ÚNICO** a cada pasaje vendido asociado a un **ASIENTO ÚNICO** para evitar inconsistencia.
- Visualizar la **LISTA DE PASAJES VENDIDOS** por cada turno.
- Visualizar la **LISTA DE ASIENTOS DISPONIBLES.**
- Al recaudador de boletería cancelar una reserva a **CLIENTES VIP.**
- Al recaudador de boletería cambiar la **FECHA DEL PASAJE** por una ocasión sin **PENALIZACIÓN**, la segunda vez tendrá una **SOBRECARGA** del 10 % del pasaje.
- Actualizará la lista de pasajeros por cada **BOLETO VENDIDO.**
- Mostrará una notificación cuando no existen **PASAJES** disponibles para un determinado turno.
- Clasificar a un nuevo **PASAJERO** con **TARIFA PREFERENCIAL.**
- Descontar el **VALOR DEL PASAJE** a los pasajeros con tarifa preferencial.
- Al **USUARIO** buscar rutas a través de los criterios: **ORIGEN DE LA CIUDAD**, **DESTINO DE LA CIUDAD** y la **FECHA**.

- Buscar las rutas con cada uno de sus **TURNOS** de acuerdo a los criterios seleccionados por el usuario.
- Mostrar la **LISTA DE TURNOS** disponibles.
- Al usuario seleccionar el **ASIENTO** en un turno específico.

Al finalizar el proceso de inspección, notará que los conceptos que finalmente se pudieron rescatar del documento anterior son los siguientes:

- Sistema. - Destino de la ciudad.	- Recaudador de boletería.	- Lista de pasajes vendidos.	- Lista de Asientos Disponibles.
- Turnos.	- Pasaje.	- Destinos.	- Rutas.
- Pasajero.	- Venta.	- Lista de pasajeros.	- Boleto.
- Capacidad máxima.	- Cupos.	- Asientos.	- Reservación.
- Identificador único.	- Asiento único.	- Bus.	- Empresa.
- Cliente VIP.	- Fecha del pasaje.	- Penalización.	- Sobrecarga.
- Boleto Vendido.	- Tarifa preferencial.	- Valor del pasaje.	- Usuario.
- Origen de la ciudad.		- Fecha.	- Lista de turnos.

4.2.2. Depuración de la lista de conceptos obtenidos.

Entendiendo que la lista de conceptos identificada durante la inspección gramatical es sólo un conjunto de conceptos candidatos, es necesario un proceso de depuración.

El proceso de depuración de la lista obtenida durante la inspección gramatical debe hacerse para eliminar términos que son sinónimos, están duplicados o se encuentren escritos de forma incorrecta y aquellos

que no tienen significado como conceptos del dominio. Para ello es necesario considerar los siguientes pasos:

- Analice las definiciones de cada término del glosario y verifique si alguna de ellas está repetida. Ello le indicará que esos términos son sinónimos y con ello necesitará llegar a un acuerdo para utilizar uno solo durante la construcción de los modelos. Un ejemplo de ello que se puede notar en dos conceptos que tienen el mismo significado son Pasaje, Boleto y Boleto Vendido, en este caso, los analistas deberán ponerse de acuerdo para seleccionar qué concepto deberá ser incluido como candidato.

- Sistema. - ~~Destino de la ciudad.~~	- Recaudador de boletería.	- ~~Lista de pasajes vendidos.~~	- Lista de Asientos Disponibles.
- Turnos.	- Pasaje.	- ~~Destinos.~~	- Rutas.
- Pasajero.	- Venta.	- Lista de pasajeros.	- ~~Boleto.~~
- Capacidad máxima.	- Cupos.	- Asientos.	- Reservación.
- Identificador único.	- Asiento único.	- Bus.	- Empresa.
- ~~Cliente VIP.~~	- ~~Fecha.~~	- Penalización.	- Sobrecarga.
- ~~Boleto Vendido.~~	- Tarifa preferencial.	- Valor del pasaje.	- ~~Usuario.~~
- ~~Origen de la ciudad~~.		- Fecha del pasaje.	- Lista de turnos.

- Convierta los términos que están en plural a singular, lo que importa en el modelo conceptual son los modelos atómicos. Muchos analistas tienden a confundir conceptos en plural con conceptos colectivos, por ejemplo, el concepto “lista de pasajeros” no es un concepto en plural sino un concepto colectivo.

- Sistema. Pasaje.	- Recaudador de boletería.	- Rutas.	- Turnos.
- Asientos.	- Lista de pasajeros.	- Pasajero.	- Venta.
- Bus.	- Reservación.	- Capacidad máxima.	- Cupos.
- Lista de Asientos Disponibles.	- Empresa.	- Identificador único.	- Asiento único.
- Valor del pasaje.	- Penalización.	- Sobrecarga.	- Tarifa preferencial.
	- Ciudad.	- Fecha del pasaje.	- Lista de turnos.

- Verifique si algún concepto es un atributo. Por ejemplo, la fecha del pasaje es información del boleto, por lo tanto, el boleto contiene información como la fecha del pasaje, hora, entre otros.

- Sistema. Pasaje.	- Recaudador de boletería.	- Rutas.	- Turnos.
- Asientos.	- Lista de pasajeros.	- Pasajero.	- Venta.
- Bus.	- Reservación.	- ~~Capacidad máxima.~~	- Cupos.
- Lista de Asientos Disponibles.	- Empresa.	- ~~Identificador único.~~	- ~~Asiento único.~~
- ~~Valor del pasaje.~~	- Penalización.	- ~~Sobrecarga.~~	- Tarifa preferencial.
	- Ciudad.	- ~~Fecha del pasaje.~~	- Lista de turnos.

- Verifique si cada concepto pertenece al dominio del problema que está tratando de solucionar. Ello le permitirá eliminar conceptos demasiados generales, innecesarios o que no tienen

ningún significado en el dominio en cuestión. En el sistema de TRANS&TURIS note que el término "sistema" es tan vago y general que no pertenece al dominio del problema. De la misma manera, "cupo" aparentemente pertenece al dominio del problema, pero sigue siendo un concepto vago y muy general, ya que podría ser interpretado como la capacidad máxima de un bus o como la capacidad disponible que existe en un bus dentro del turno, etc.

- ~~Sistema.~~	- Recaudador de boletería.	- Ruta.	- Turno.
- Pasaje.	- Lista de pasajeros.	- Pasajero.	- Venta.
- Asiento.	- Reservación.	- ~~Cupo.~~	- Bus.
- ~~Empresa.~~	- Lista de Asientos Disponibles.	- Penalización.	- Tarifa preferencial.
- Ciudad.	- Lista de turnos.		

- Elimine acciones, reportes (elementos físicos) o papeles que cumplen los usuarios.

- ~~Recaudador de boletería~~.	- ~~Lista de Asientos Disponibles~~.	- Turno.	- Pasaje.
- Lista de pasajeros.	- Pasajero.	- ~~Venta~~.	- Asiento.
- Reservación.	- Bus.	- Penalización.	- Tarifa preferencial.
- Ciudad.	- Ruta.	- ~~Lista de turnos~~.	

A continuación, se presenta la lista de conceptos una vez que se ha depurado.

Lista de frases sustantivas:

- Ruta.	- Lista de pasajeros.	- Reservación.	- Ciudad.
- Turno.	- Pasajero.	- Bus.	- Tarifa
- Pasaje.	- Asiento.	- Penalización.	Preferencial.

Hasta este punto, una vez realizada la depuración de conceptos, es una buena práctica actualizar la descripción de los requisitos de acuerdo al contexto del dominio del problema que se depuró. Continuando con el ejemplo los requisitos funcionales quedarían de la siguiente manera:

El sistema deberá permitir:

- Al recaudador de boletería administrar destinos.
- Al recaudador de boletería registrar nuevos pasajeros.
- Al recaudador de boletería administrar turnos para rutas.
- Registrar los pasajes vendidos en la lista de pasajeros de cada turno.
- Imprimir o descargar los pasajes vendidos al pasajero.
- Evitar la venta duplicada de pasajes en un mismo turno.
- Realizar la reservación de pasaje para un turno específico.
- Controlar la capacidad máxima de asientos de cada bus de la empresa.
- Asignar un identificador único a cada pasaje vendido asociado a un asiento único para evitar inconsistencia.
- Visualizar la lista de pasajes vendidos por cada turno.
- Visualizar la lista de asientos disponibles o que no posean pasajes asociados.
- Al recaudador de boletería cancelar una reserva a pasajeros VIP.
- Al recaudador de boletería cambiar la fecha del pasaje por una ocasión sin penalización, la segunda vez tendrá una sobrecarga del 10 % del pasaje.
- Actualizar la lista de pasajeros por cada pasaje vendido.
- Mostrará una notificación cuando no existen pasajes disponibles para un determinado turno.
- Clasificar a un nuevo pasajero con tarifa preferencial.

- Descontar el valor del pasaje a los pasajeros con tarifa preferencial.
- Al usuario buscar rutas a través de los criterios: origen de la ciudad, destino de la ciudad y la fecha.
- Buscar las rutas con cada uno de sus turnos de acuerdo a los criterios seleccionados por el usuario.
- Mostrar la lista de turnos disponibles.
- Al usuario seleccionar asiento(s) en un turno específico para la compra de pasajes.

4.2.3. Representación de los conceptos en UML (Lenguaje Unificado para el Modelado).

En esta sección no se definirán los conceptos de UML ni de sus diagramas. Se representará cada concepto usando las diferentes representaciones para el modelo del dominio. En UML cada concepto es una clase que se representa a través de un rectángulo. A continuación, se presenta el modelo conceptual preliminar obtenido del ejemplo de TRANS&TURIS:

Figura 13. Modelo conceptual preliminar del sistema de boletería de TRANS&TURIS.

4.2.4. Identificación de relaciones entre los conceptos.

En realidad, las cosas del mundo real no existen aisladas unas de otras. Durante la construcción del modelo conceptual es necesario encontrar las relaciones existentes entre los conceptos que hasta ahora se han identificado.

Las relaciones entre objetos de cierta clase (concepto) son importantes debido a que permiten definir cómo los objetos de cada clase interactúan con los de otras clases. Las relaciones que se aplican son de: asociación simple, agregación, composición o herencia. Un consejo práctico es no preocuparse que tipo de relación existe entre cada concepto, si el analista no está seguro del tipo de relación, aplicar la relación de asociación simple, ya que la idea principal del modelo conceptual es visualizar los conceptos del dominio del problema y como se encuentran interrelacionados; y además que el diagramado sea lo más rápido posible.

En el caso TRANS&TURIS, se puede encontrar entre otras las siguientes relaciones:

- Un turno ***tiene*** una ruta (asociación).
- Un pasaje ***tiene*** una reservación (asociación).
- Un bus ***realiza*** viaje en una ruta determinada (asociación).
- Un turno ***tiene*** una lista de pasajeros (asociación).
- Un bus ***es un*** vehículo (herencia).
- Un pasajero ***es una*** persona (herencia).
- Un pasaje ***es parte de*** la lista de pasajeros (agregación).
- Un pasajero ***tiene*** una tarifa preferencial (asociación).
- Un pasajero ***adquiere*** un pasaje (asociación).
- Un pasaje ***puede tener*** penalización (asociación).
- Un asiento ***es parte de*** un bus (agregación).

En el listado anterior, la herencia “bus es un vehículo” se infiere de la observación que existe en la realidad debido a que “Bus” es un tipo de “Vehículo” y se da la posibilidad a que se pueda agregar al dominio de la aplicación más tipos de vehículos.

El siguiente paso, indica el diagrama de clases que representa parte del modelo conceptual de la aplicación para boletería, donde se visualiza los conceptos interrelacionados que intervienen en el negocio y que están acorde a la especificación de requisitos funcionales.

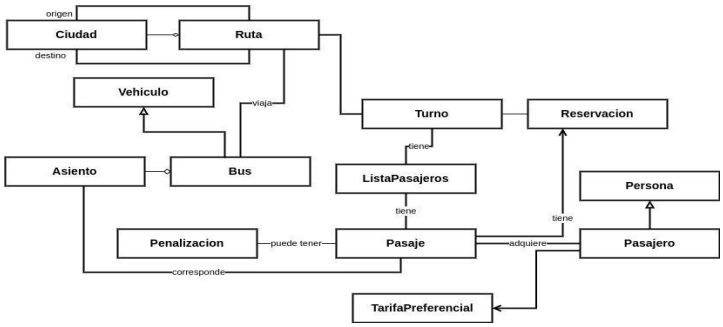


Figura 14. Diagrama que representa parte del modelo conceptual del sistema de boletería.

4.2.5. Inclusión de atributos en el modelo conceptual, transformándolo al modelo de dominio.

Durante el proceso de inspección gramatical, los analistas encontrarán una cantidad de conceptos que, aunque podrían ser planteados como clases, en realidad no tienen un significado fuera de otro concepto mayor que los contengan.

Por ejemplo, considere los conceptos nombre, fecha de nacimiento, número máximo de pasajeros, entre otros, estos conceptos debieron haber sido identificados durante la actividad de la inspección gramatical como conceptos potenciales, pues podrían cumplir con toda seguridad

las reglas citadas para identificar a las clases. Es decir, ser sustantivos, pertenecer al dominio, etc.

Sin embargo, todos los ejemplos citados no tienen un significado sin que exista un concepto mayor que los contenga, por ejemplo, considere el concepto nombre, este concepto cumple con ser sustantivo, puede ser parte del dominio en estudio y está es su forma singular, este concepto califica perfectamente como una clase candidata; pero, no tiene sentido sin que exista un concepto mayor como el de persona, en otras palabras, nombre es parte de la información de persona. Como regla general, cualquier concepto cuya existencia es dependiente de otro se conoce como atributo.

Los atributos no son otra cosa que características que permiten describir a una clase. La identificación de los atributos de cada concepto es una de las últimas actividades que se debe cumplir para finalizar la construcción del modelo conceptual del sistema.

Existen algunos criterios que deberían ser tomados en cuenta para poder colocar atributos en cada una de las clases.

- Algunos atributos se encontraron durante la inspección gramatical, pero que no tiene suficiente "peso" para ser considerados clases. Por ejemplo, "fecha de nacimiento", es un atributo que como sustantivo forma parte del dominio pero que no tiene identidad propia como para que pueda ser definido como clase.
- Los atributos deberían ser valores puros de datos, es decir no deben ser estructuras compuestas. Esto implica que los atributos puedan almacenar valores atómicos de información, por lo general de algún tipo numérico o carácter.
- Los atributos generalmente se pueden distinguir durante la inspección gramatical como sustantivos posesivos, es decir

puede preguntarse si uno de los sustantivos identificados permite describir o es una característica de otro de los conceptos encontrados. Por ejemplo, aunque la fecha de nacimiento se logró distinguir como un sustantivo que forma parte del dominio, lo correcto es considerar que ese concepto puede ser de mejor forma como fecha de nacimiento del pasajero, en cuyo caso ese sustantivo se transforma de ser un concepto a un atributo de la clase pasajero.

- No debe incluirse bajo ningún motivo atributos que se sabe no son útiles en el dominio en cuestión, considere el caso de un pasajero en el dominio de boletería, el atributo estado civil no tiene mucha importancia para el dominio del problema. Debe tratar de mantener el modelo lo más simple posible y no incluir elementos extraños o ajenos al dominio.

Se muestra el diagrama de clases donde se incluyen los atributos identificados en el modelo conceptual de la aplicación de boletería para TRANS&TURIS:

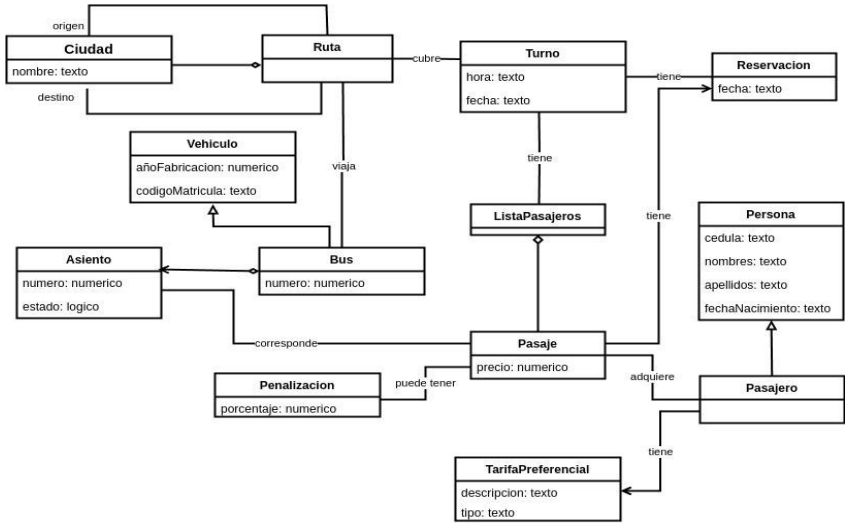


Figura 15. Modelo de dominio de TRANS&TURIS con algunos atributos.

4.2.6. Inclusión de la visibilidad, multiplicidad y navegabilidad en el modelo del dominio.

4.2.6.1. Visibilidad (Accesibilidad)

La idea básica que se debe entender por visibilidad es que cualquier atributo o método de una clase es público o privado. La visibilidad es aquella característica a través de la cual un objeto muestra u oculta sus atributos al resto de objetos.

- **Visibilidad Pública (+)**: Un atributo o método de una clase que disponga visibilidad pública puede ser accedida desde cualquier otro objeto.
- **Visibilidad Privada (-)**: Un atributo o método de una clase que se defina como privado no puede ser accedido por ningún otro objeto.
- **Visibilidad Protegida (#)**: Un atributo o método de una clase que disponga visibilidad protegida puede ser accedida por la clase que lo define o por cualquier subclase de esta.

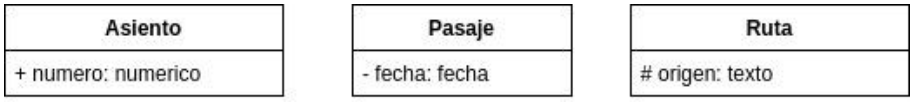


Figura 16. Atributos con visibilidad pública, privada y protegida en UML.

4.2.6.2. Multiplicidad

Otro aspecto importante es la multiplicidad, que es una característica en la que se definen cuantos objetos de cada clase intervienen en la relación de asociación. La multiplicidad puede expresarse de las siguientes formas:

Tabla 19. Tipos de multiplicidad en UML y su significado.

Multiplicidad	Significado
0..*	Cero o más elementos (opcional).
1..*	Uno o más elementos (obligatoria).
1..n	Desde uno hasta n elementos (n>1).
N	Exactamente n elementos (n>1).
0..1	Cero o uno.
1	Exactamente 1.

4.2.6.3. Navegabilidad

La navegabilidad es una característica de una relación de asociación en el cual un objeto de una clase necesita tener visibilidad de un objeto de otra clase que interviene en la relación. Puede darse el caso de que la navegabilidad sea de ambos lados de la relación, esto quiere decir que, si dos clases A y B tienen una relación de asociación, entonces A necesita tener visibilidad de los objetos de la clase B y B necesita tener visibilidad de los objetos de la clase A. En otros casos, solo se necesita tener visibilidad en un sentido, en este caso se dice que el mismo ejemplo de las clases A y B, A necesita tener visibilidad de los objetos de la clase B, pero B no necesita tener visibilidad de los objetos de la clase A. La navegabilidad se representa a través de la inclusión de las flechas en el sentido de la navegabilidad, por ejemplo:

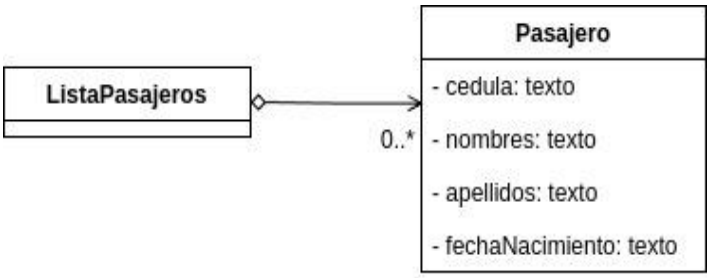


Figura 17. Ejemplo de navegabilidad en un solo sentido.

Por el contrario, en la navegabilidad en ambos sentidos se omiten las flechas.

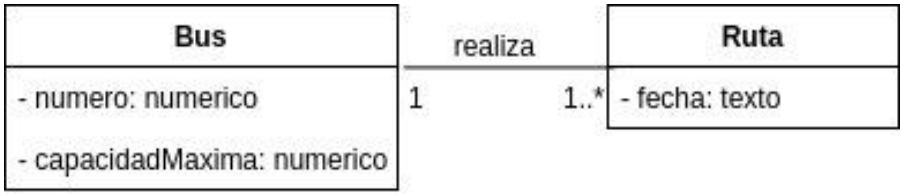


Figura 18. Ejemplo de navegabilidad de ambos sentidos.

Finalmente, analizando el diagrama del ejemplo de TRANS&TURIS y aplicando la visibilidad, multiplicidad y la navegación se define el modelo del dominio:

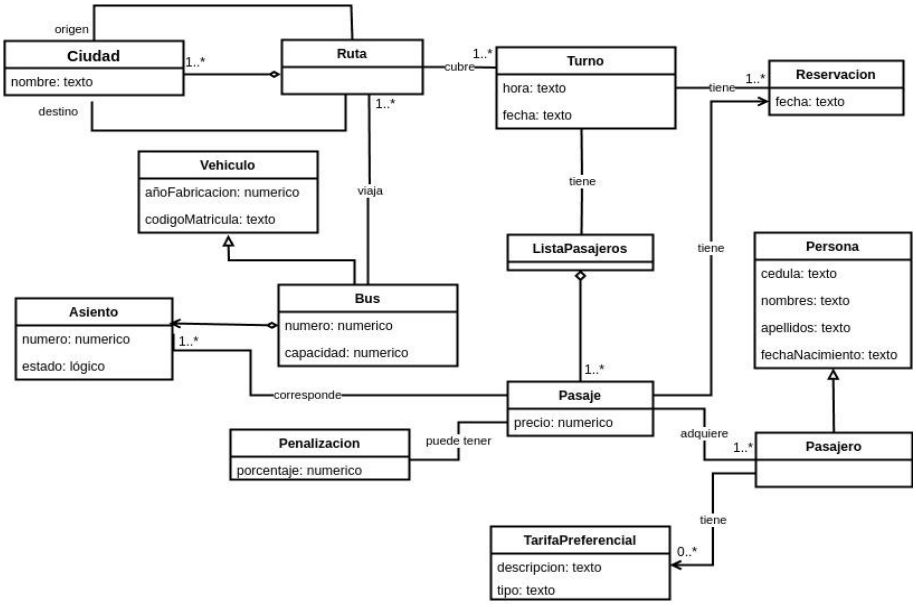


Figura 19. Modelo de dominio final de la aplicación TRANS&TURIS.

Otro consejo práctico es no preocuparse o perder tiempo en la navegabilidad, si el analista no está seguro del tipo de navegabilidad, aplicar la navegabilidad en ambos sentidos, como ya se ha mencionado la idea principal del modelo de dominio es visualizar los conceptos del dominio del problema y como se encuentran interrelacionados; y además que el diagramado sea lo más rápido posible, con lo cual ya se está cumpliendo con este objetivo.

5. Modelo de Casos de Uso.

El modelado de Casos de uso brinda un enfoque para describir los requisitos funcionales del sistema en términos de actores y casos de uso, permitiendo crear un diseño razonable a partir de ellos, por ende, son fundamentales tanto para la fase de análisis o diseño preliminar porque sirven como base para: descubrir requisitos, corregir aquellos requisitos que contienen errores, para aclarar los requisitos que posean ambigüedades o confusión y para actualizar el modelo del dominio (Fuentes, 2011, p. 70).

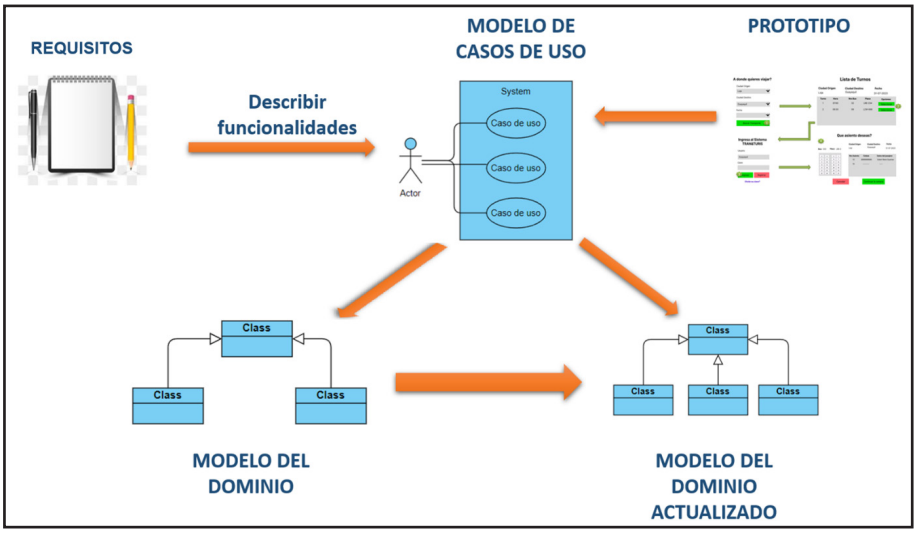


Figura 20. Representación del modelo de casos de uso.

En sí, el modelo de casos de uso es un proceso de análisis complejo que involucra a todo el grupo de analistas que les ayuda a responder las siguientes preguntas: (i) ¿Qué intentan hacer los usuarios del sistema? y (ii) ¿Cuál es la experiencia del usuario?, debido a que una gran cantidad de lo que hace el software está dictada por la forma en que los usuarios deben interactuar con él y su meta es generar lo siguiente:

1. Diagrama de casos de uso.
2. Narración o descripción de casos de uso.

El diagrama de casos de usos nos permite capturar el comportamiento deseado del sistema en desarrollo sin tener que especificar los detalles cómo se implementa este comportamiento; se utiliza la nomenclatura de UML para visualizar gráficamente la funcionalidad o escenarios que tendrá el sistema a través del caso de uso, los actores que participan en esos escenarios, relaciones y la frontera del sistema, de tal manera que satisfagan los requisitos funcionales.

A continuación, se indican algunas definiciones básicas para poder dibujar un diagrama de casos de uso correctamente.

El **actor** representa cualquier **entidad externa** que necesita interactuar con el sistema a través de un **rol específico** que pueda desempeñar; pueden ser:

- Primarios, son aquellos actores que interactúan frecuentemente con el caso de uso (sistema) para lograr un objetivo o explotar su funcionalidad.

- Secundarios, son aquellos actores que brindan soporte o asistencia al sistema para que el caso de uso logre su objetivo.

Figura 21. Representación de un actor.

Caso de uso representa la funcionalidad que debe cumplir el sistema, la cual debe cubrir uno o más requisitos funcionales. Nos indica el uso que tendrá el sistema que se está diseñando a través de la definición de la secuencia de interacciones entre las acciones de uno o más actores y comportamiento que poseerá el sistema desde el punto de vista del usuario para lograr un objetivo. Los analistas de sistemas utilizan los casos de uso para verificar el cumplimiento de los requisitos y describir las funcionalidades del sistema evitando los detalles de implementación.

El nombre de un CU debe comenzar por un **verbo en infinitivo y luego tener palabras complementarias** definidas en el contexto del problema que aclaren el propósito del caso de uso.

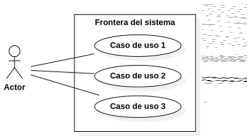


Figura 22. Representación de caso de uso.

Frontera, representa el alcance que va poseer el sistema, es decir que es interno y externo al comportamiento del sistema.

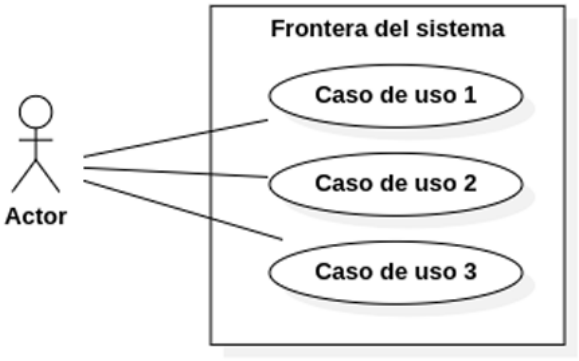


Figura 23. Representación de la frontera del sistema.

En esta sección sólo se detallaron algunos conceptos generales de UML para dibujar el diagrama de casos de uso, ya que la idea principal de la sección es enfocarse en cómo identificar los requisitos para representarlos en casos de uso.

La primera parte es documentar la lista de requisitos final a través de un documento de especificación de requisitos. Como se menciona en apartados anteriores se ha elegido el formato de lista, por ser: simple, directo, ágil y verificable, como se detalla a continuación:

Tabla 23. Requisitos funcionales de sistema TRANS&TURIS.

El sistema permitirá:

Código	Descripción
RF01	A los usuarios autenticarse en el sistema.
RF02	Al administrador desactivar la cuenta de usuario.
RF03	Al **RECAUDADOR DE BOLETERÍA** administrar destinos y rutas.
RF04	Al recaudador de boletería registrar nuevos **PASAJEROS.**
RF05	Al recaudador de boletería administrar **TURNOS** para **RUTAS.**
RF06	Registrar los **PASAJES** vendidos en la **LISTA DE PASAJEROS** de cada **TURNO.**
RF07	Imprimir o descargar los pasajes vendidos al **PASAJERO.**
RF08	Evitar la **VENTA** duplicada de pasajes para un turno.
RF09	Al **USUARIO** / recaudador de boletería reservar pasajes para un turno específico.
RF10	Asignar un **IDENTIFICADOR ÚNICO** a cada **PASAJE** vendido asociado a un **ASIENTO ÚNICO** para evitar inconsistencia.
RF11	Controlar la capacidad máxima de asientos de cada bus.

Código	Descripción
RF12	Al usuario / recaudador de boletería visualizar la lista de pasajes vendidos por cada turno.
RF13	Al usuario / recaudador de boletería visualizar la **LISTA DE ASIENTOS** disponibles o que no posean pasajes asociados.
RF14	Al recaudador de boletería cancelar una reserva a **PASAJEROS VIP.**
RF15	Al recaudador de boletería cambiar la **FECHA DEL PASAJE** por una ocasión sin **PENALIZACIÓN**, la segunda vez tendrá una **SOBRECARGA** del 10 % del pasaje.
RF16	Actualizar la lista de pasajeros por cada **PASAJE VENDIDO.**
RF17	Mostrará una notificación cuando no existen pasajes disponibles para un determinado turno.
RF18	Clasificar a un nuevo pasajero con **TARIFA PREFERENCIAL.**
RF19	Calcular el **VALOR DEL BOLETO** de acuerdo a la clasificación del pasajero / tarifa preferencial.
RF20	Al usuario buscar rutas a través de los criterios: **ORIGEN DE LA CIUDAD**, **DESTINO DE LA CIUDAD** y la **FECHA**.
RF21	Buscar las rutas con cada uno de sus turnos de acuerdo a los criterios seleccionados por el usuario.
RF22	Mostrar la **LISTA DE TURNOS** disponibles.
RF23	Al usuario seleccionar **ASIENTO(S)** en un turno específico para la compra de pasajes.

5.1. Identificación de casos de uso.

El primer paso en el modelado de casos de uso es identificar los casos de uso y sus actores. Se realiza un análisis gramatical de las principales funcionalidades que va a poseer el sistema según la lista de requisitos funcionales. Cada caso de uso debe agrupar uno o varios requisitos funcionales.

Tabla 24. Lista de Casos de Uso

Actores	Casos de uso	ID UC	Ref. Requisitos
Todos.	Ingresar al sistema.	UC01	RF01
Administrador.	Desactivar Usuarios.	UC02	RF02
Recaudador de boletería.	Administrar destinos y rutas.	UC03	RF03
Todos.	Buscar rutas y turnos.	UC04	RF20, FR22
Recaudador de boletería.	Vender pasaje.	UC05	RF06, RF07, RF08, RF10, RF13, RF16, RF19
Recaudador de boletería.	Canjear pasaje.	UC06	RF15
Recaudador de boletería.	Administrar pasajeros.	UC07	RF04, RF18
Usuario / recaudador de boletería.	Reservar pasaje.	UC08	RF09, RF14
Recaudador de boletería.	Administrar turnos.	UC09	RF05, RF11

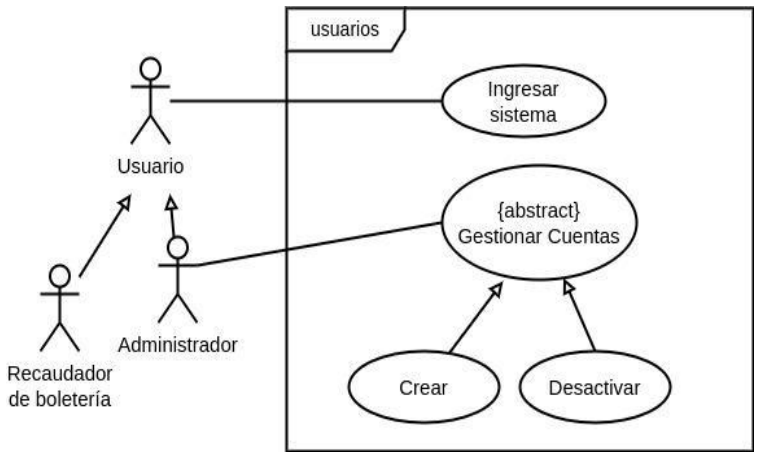


Figura 24. Diagrama de casos de uso de gestión de usuarios.

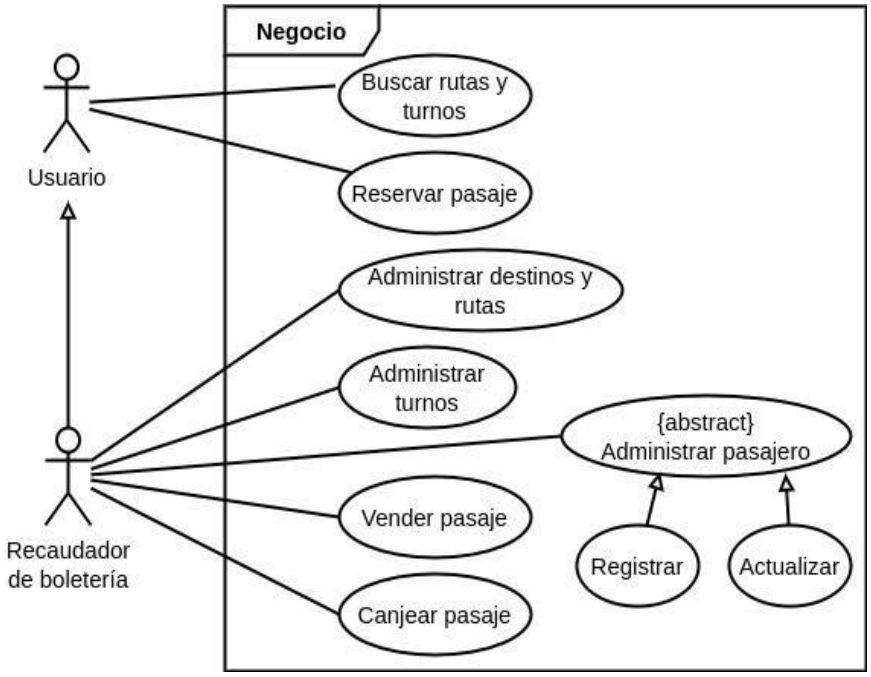


Figura 25. Diagrama de casos de uso del negocio.

5.2. Especificación de Casos de Uso.

Un caso de uso es considerado una técnica de elicitación en la ingeniería de requisitos para la captura de información. La información recopilada debe ser documentada por los analistas a través de la especificación de casos de uso que contiene información respecto al comportamiento de un caso de uso, por ejemplo, el nombre, identificador, actores y los diferentes escenarios.

La narración de los escenarios en un caso de uso se realiza con el apoyo de un prototipo o sin él. El prototipo garantiza calidad en la especificación porque se realiza un detalle preciso de las acciones del actor y del sistema.

El proyecto (*Marco De Desarrollo De La Junta De Andalucía | MADEJA*) señala que los casos de uso al pertenecer a las técnicas de elicitación deben ser documentadas, y da algunas recomendaciones para especificar casos de uso y evitar problemas de interpretación:

- El caso de uso debe describir sólo comportamiento observable externamente, sin entrar en la funcionalidad interna del sistema.
- La invocación de unos casos de uso desde otros casos de uso (inclusión, o extensión), sólo debe usarse como un mecanismo para evitar repetir una determinada secuencia de pasos que se repite en varios casos de uso. Nunca debe usarse para expresar posibles menús de la interfaz de usuario.
- Evitar narraciones con estructuras condicionales en la descripción del caso de uso.
- Todos los casos de uso deben estar descritos al mismo nivel de detalle.

Hay varios tipos de plantilla al especificar un caso de uso. En la siguiente tabla se define las partes que contiene la plantilla para la especificación de casos de uso:

Tabla 25. Formato de especificación de caso de uso.

Nombre de caso de uso	El nombre del caso de uso expresa una funcionalidad de alto nivel que debe ser narrado con verbo en infinitivo.
ID de caso de uso	Es el identificador único del caso de uso que suele iniciar con UC o CU seguido de un número.
Prioridad	La prioridad será alta, media o baja dependiendo de la importancia de implementar el caso de uso en la primera, segunda, tercera versión del sistema.
Requisitos	Es la lista de requisitos que el caso de uso resolverá.
Actor primario	Son usuarios del sistema que ejecutan el caso de uso.
Descripción	Detalla el objetivo y el rol del caso de uso.
Precondición	Son acciones o estados que deben ejecutarse antes que empiece la funcionalidad de un caso de uso.

Curso Normal de Eventos	
Consiste en la narración secuencial de pasos donde indica las interacciones que realiza el actor y las respuestas que debe realizar el sistema sin detalles de implementación con la finalidad de cumplir el escenario especificado, por lo tanto, describen el comportamiento que tendrá el sistema en ese escenario y proporcionan un medio para que los desarrolladores, los usuarios finales del sistema y los expertos del dominio lleguen a una comprensión común del sistema.	
Acciones del Actor	**Acciones del Sistema**
Son las acciones realizadas por el actor de sistema.	Son las acciones que ejecuta el sistema internamente y no son ocultas al actor del sistema. El resultado de estas acciones es visible al actor.
Curso Alterno de Eventos	
Es una excepción que sucede en algún paso del curso normal de eventos alterando el resultado que el actor espera.	
Postcondición	Es el resultado de las acciones luego de ejecutar el caso de uso.

Cabe señalar que la redacción del Curso Normal de Eventos se puede realizar de varias formas:

- **Párrafo**, la que consiste en describir el flujo de eventos y las interacciones entre un actor y el sistema de manera continua y detallada, sin segmentar la información en secciones o pasos;
- **Secuencial**, que consiste en desglosar el flujo de eventos de un caso de uso en una serie de pasos bien definidos y numerados. Este enfoque permite describir cada interacción entre los actores y el sistema de manera ordenada y fácil de seguir. Este formato enfatiza la secuencia lógica y detallada de las acciones, facilitando la comprensión del flujo de eventos y posibles variaciones o alternativas.
- **Secuencial dividiendo "acción del actor - respuesta del sistema",** la misma que consiste en descomponer el flujo de eventos en una secuencia de pasos alternados, donde cada acción realizada por el actor es seguida por la correspondiente respuesta del sistema. Este formato permite una clara separación de las

interacciones, mostrando cómo el sistema reacciona ante cada acción del actor en un ciclo de "acción-respuesta". Es un enfoque muy estructurado que facilita la comprensión precisa de cómo el sistema interactúa con los actores involucrados. (Como se indica, en tabla 25).

5.3. Ejemplos de Especificación de Casos de Uso.

Con el propósito de describir varias formas de narrar un caso de uso, en esta sección se detallan tres ejemplos de especificación de casos de uso con su respectivo prototipo.

5.3.1. Narración Caso de Uso: Registrar pasajero.

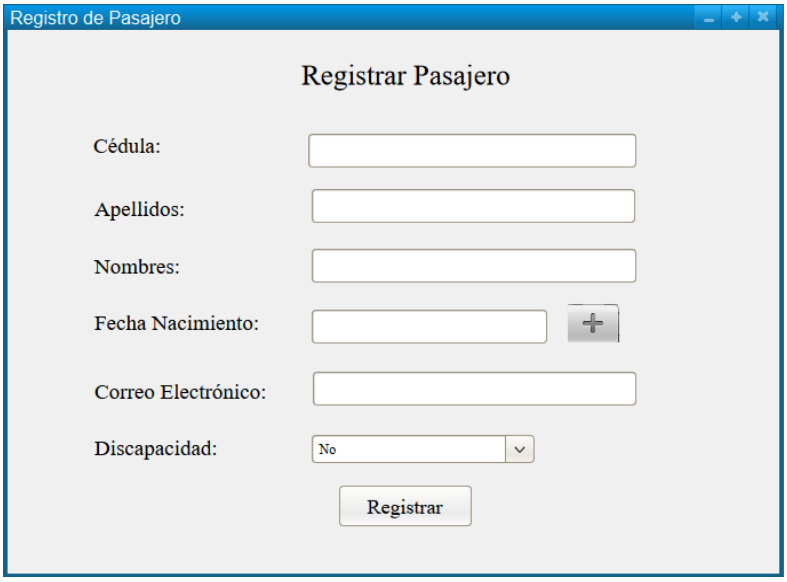


Figura 26. Prototipo Registrar pasajero.

En la siguiente tabla se presenta la narración del caso en base al prototipo indicado para registrar pasajeros.

Tabla 23. Especificación de Caso de Uso “Registrar pasajero”

Autor/es: Wilman Chamba.	**Fecha:** 29-06-2023. **Versión** 1.0.
Nombre de caso de uso	Registrar pasajero.
ID de caso de uso	UC07.
Prioridad	Alta.
Fuente / Referencia / requisitos	RF04, RF18.
Actor primario	Recaudador de boletería.
Descripción	El caso de uso case registra los datos de un nuevo pasajero.
Precondición	Que el actor se encuentre autenticado en el sistema. Que el actor haya ingresado a la pantalla **Registrar pasajero**.
Curso Normal de Eventos	
Acciones del Recaudador de boletería	**Acciones del Sistema**
1. Ingresa los datos del pasajero (cédula, apellidos y nombres, fecha de nacimiento y correo electrónico) y selecciona el tipo de discapacidad en la pantalla **Registrar pasajero**. 2. Selecciona el botón **Registrar**.	3. Valida que los datos obligatorios del pasajero no se encuentran vacíos en la pantalla **Registrar pasajero.** 4. Calcula la edad del pasajero en base a la fecha de nacimiento. 5. Clasifica el tipo de pasajero (normal, tercera edad, menor de edad) de acuerdo a su edad. 6. Guarda al pasajero. 7. Muestra un mensaje de confirmación “Pasajero registrado".
Curso Alterno de Eventos	
A. VALIDACIÓN DE DATOS OBLIGATORIOS. A3. El sistema muestra un mensaje de aviso "Datos obligatorios requeridos".	
Postcondición	Pasajero registrado en el sistema.

5.3.2. Narración Caso de Uso: Vender pasaje.

Figura 27. Prototipo Lista de turnos disponibles.

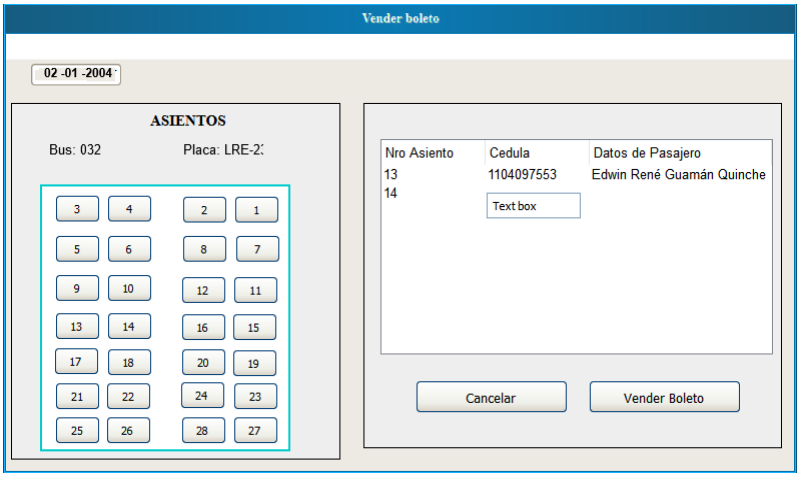


Figura 28. Prototipo Vender Pasajes.

En la siguiente tabla se presenta la narración del caso en base al prototipo indicado para Vender Pasaje.

Tabla 24. Especificación caso de uso Vender pasaje.

Autor/es: Wilman Chamba.	**Fecha:** 30 de junio de 2023. **Versión** 1.0.
Nombre de caso de uso	Vender pasaje.
ID de caso de uso	UC05.
Prioridad	Alta.
Fuente / Referencia / requisitos	RF06, RF07, RF08, RF10, RF13, RF16, RF19.
Actor primario	Recaudador de boletería.
Descripción	Además, se selecciona un pasaje para asignar a un pasajero.
Precondición	El actor se haya autenticado en el sistema. El actor haya buscado un conjunto de rutas. El actor haya ingresado a la pantalla **Lista de turnos disponibles**.
Curso Normal de Eventos	
Acciones de Recaudador de Boletería	**Acciones del Sistema**
1. Selecciona un turno en la opción escoger de la pantalla **Lista de turnos disponibles**.	2. Busca los asientos disponibles del turno seleccionado. 3. Carga y despliega la distribución de los asientos del bus en la pantalla **Vender Boleto**. 4. Pinta los asientos vendidos de color rojo, los asientos reservados de color verde, y sin color los asientos disponibles para la venta en la pantalla **Vender Boleto.**
5. Selecciona el o los asientos en la sección Asientos de la pantalla **Vender Boleto.** 7. Ingresa la cédula en el campo de texto cédula de la tabla **Datos Pasajeros**.	6. Carga los asientos en la tabla **Datos Pasajeros.** 8. Invoca al caso de uso **Buscar Pasajero**. 9. Carga los datos del pasajero en la tabla **Datos Pasajeros** de la tabla.

10. Selecciona la opción Vender Boleto.	11. El sistema calcula el valor del boleto de acuerdo al tipo de tarifa del pasajero. 12. Muestra un mensaje de confirmación.
Cursos Alternativos	
A. Pasajero no registrado.	
	A.9 Muestra la pantalla registrar pasajero nuevo e invoca al caso de uso **Registra Pasajero**.
Postcondición	El pasaje ha sido asignado a un pasajero.

5.3.3. Narración Caso de Uso: Buscar Rutas y Turnos.

Figura 29. Prototipo Buscar ruta.

Figura 30. Prototipo Lista de Turnos disponibles.

Tabla 25. Especificación Caso de Uso Buscar rutas y turnos

Autor/es: Wilman Chamba.	**Fecha** 27 de julio de 2023. **Versión** 1.0.
Nombre de caso de uso	Buscar rutas y turnos.
ID de caso de uso	UC04.
Prioridad	Alta.
Fuente / Referencia / requisitos	RF20, FR22.
Actor primario	Usuario (abstracto, Recaudador de boletería, Administrador.
Descripción	El caso de uso permite buscar rutas de acuerdo a los criterios de búsqueda: origen, destino y fecha; para desplegar una lista de turnos disponibles.
Precondición	El usuario se haya autenticado en el sistema. El usuario haya ingresado a la pantalla **Buscar Rutas.**

Curso Normal de Eventos	
Acciones de Recaudador de Boletería	**Acciones del Sistema**
1. Selecciona o ingresa una ciudad de origen.	2. Busca las ciudades de destino de acuerdo con la ciudad de origen seleccionada. 3. Carga la lista de ciudades de destino en la lista de despliegue Destino de la pantalla **Buscar Rutas.**
4. Selecciona la ciudad de destino.	
5. Ingresa la fecha.	6. Busca los turnos de acuerdo a la ruta y la fecha ingresada. 7. Carga la lista de turnos en la interfaz **Lista de Turnos Disponibles.**
Postcondición	Turnos disponibles para la ruta.

En las siguientes figuras se establece un resumen visual con las pautas sugeridas para realizar un adecuado modelamiento de casos de uso y descripción de estos.

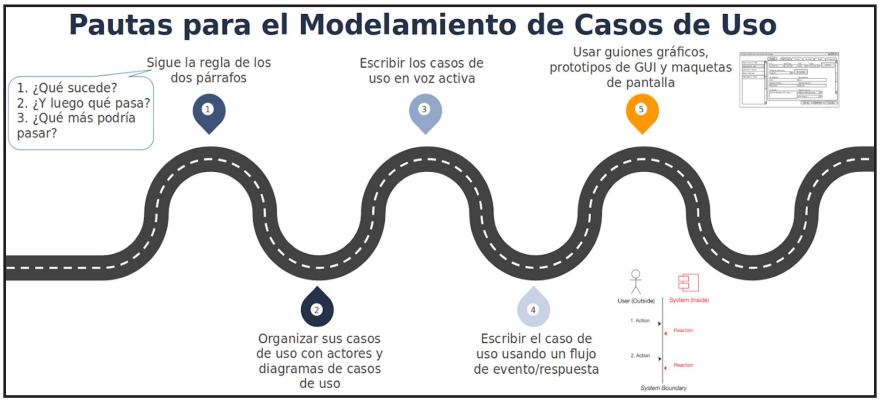


Figura 31. Pautas para el modelamiento de casos de uso, parte 1.

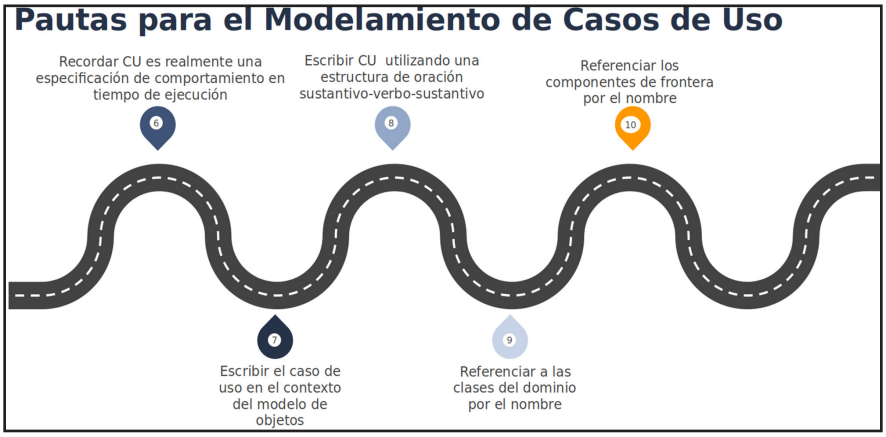


Figura 32. Pautas para el modelamiento de casos de uso, parte 2.

En resumen, los autores han presentado una de las formas prácticas para llevar a cabo la actividad de **especificación**, que es una de las cuatro actividades fundamentales en el proceso de desarrollo de software: **especificación, desarrollo, validación y evolución**. Esta actividad abarca el **análisis, diseño y arquitectura del software**, y se aplica en la mayoría de las metodologías de desarrollo mediante el **levantamiento de requisitos**. A lo largo de esta obra, se ha abordado el **modelado de requisitos**, que incluye las técnicas de elicitación, el **modelado del dominio**, y el **modelado de casos de uso**, siendo este último un puente fundamental entre el análisis de requisitos y el diseño de la arquitectura, facilitando una transición clara y estructurada entre ambos.

Bibliografía

Ambler, S. W. (2010). *The Object Primer: Agile Model-Driven Development with UML 2.0*. Cambridge University Press. https://doi.org/10.1017/CBO9780511584077

Doug Rosenberg and Matt Stephens, Use Case Driven Object Modeling with UML - Theory and Practice, ISBN-13 (pbk): 978-1-59059-774-3, ISBN-10 (pbk): 1-59059-774-5, Apress, 2007, Second Edition

Fernandes, J. M., & Machado, R. J. (2016). Requirements in engineering projects. Cham: Springer International Publishing.

Fuentes, M. D. C. G. (2011). MATERIAL DIDÁCTICO NOTAS DEL CURSO ANÁLISIS DE REQUERIMIENTOS.

González, J. E. R., Condori, F. R. H., Flores, R. S. P., & Adrianzén, D. J. F. (2021). INGENIERÍA DE REQUERIMIENTOS un enfoque práctico. Savez Editorial.

Gottesdiener, E. (2005). The Software Requirements: Memory Jogger: a Pocket Guide to Help Software and Business Teams Develop and Manage Requirements. GOAL/QPC.

Hull, E., Jackson, K., & Dick, J. (2005). Requirements engineering in the solution domain (pp. 109-129). Springer London.

Koelsch, G. (2016). Requirements writing for system engineering (p. 428). Berkeley: Apress.

Lluvia de ideas (Técnicas Participativas). (2012, septiembre 14). Piense algo y sienta hondo. Retrieved February 23, 2023, from

https://www.pienseash.com/2012/09/lluvia-de-ideas-tecnicas-participativas.html

Mannio, M. (2010). Requirements elicitation using a combination of prototypes and scenarios. *International Conference on Advanced Software Engineering and Its Applications*, 7. 10.1007/978-3-642-17578-7_4

Marco de Desarrollo de la Junta de Andalucía | MADEJA. (n.d.). Junta de Andalucía. Retrieved March 7, 2023, from https://www.juntadeandalucia.es/servicios/madeja/

NIELSEN, J. (2014, July 30). Prototipos de Papel (Paper Prototyping) | Curso de Interacción Persona-Ordenador. Curso de Interacción Persona-Ordenador. Retrieved February 23, 2023, from https://mpiua.invid.udl.cat/prototipos-de-papel-paper-prototyping/

Piattini Velthuis, M. G. (1996). Análisis y Diseño de Aplicaciones Informáticas de Gestión. Ra-Ma, Libreria y Editorial Microinformática.

Silva, R., Panach, J., & Distante, D. (2020). Requirements elicitation methods based on interviews in comparison: A family of experiments. Information and Software Technology, 126(Elsevier), 32. https://doi.org/10.1016/j.infsof.2020.106361

Warfel, T. Z. (2009). Prototyping: A Practitioner's Guide. Rosenfeld Media.

Washizaki, H. An Overview of the SWEBOK Guide Version 4.0.

Wong, S. (2017). Análisis y requerimientos de software: manual autoformativo interactivo (Primera ed.). Universidad Continental.

Allan D, Barbara H.W., David T. System Analysis & Design An Object-Oriented Approach with UML. 5th Edition. Wiley. 2015. ISBN 978-1-118-80467-4

Timothy C., Robert L. Object-Oriented Software Engineering Practical Software Development using UML and Java. 2th Edition. McGraw-Hill Education. 2005. ISBN 0-07-70109082

Anónimo. (n.d.). Casos de Uso · Libro de Desarrollo de Software. Retrieved March 6, 2023, from https://rcasalla.gitbooks.io/libro-desarrollo-de-software/content/libro/temas/t_requerimientos/req_casosuso.html

Universidad Nacional de Loja

Análisis y diseño preliminar de software basado en el modelado de requisitos y casos de uso. Un enfoque teórico y práctico

En los años 60, el avance del hardware impulsó a un crecimiento exponencial de la demanda de software, pero la falta de estándares, metodologías y definición de requisitos desató la conocida 'Crisis del Software'. Frente a esta problemática, el libro centra sus esfuerzos en resaltar la importancia crucial del modelado de requisitos en el proceso de desarrollo de software.

La obra inicia con la contextualización del desarrollo de software y subrayando la necesidad de abordar este proceso de manera estructurada. A continuación, profundiza el modelado de requisitos, diferenciando entre funcionales y no funcionales, proporcionando pautas detalladas para la redacción, validación y formalización en un documento de especificación de requisitos (DER). Las siguientes secciones presentan: técnicas esenciales para la obtención de requisitos, fundamentales para descubrir, identificar y validar necesidades; el modelado conceptual orientado a objetos, destacando el uso de buenas prácticas y diagramas de clases de UML; y, finalmente, se aborda el diseño preliminar a través del modelado de casos de uso, destacando su función en la validación y la identificación de nuevos requisitos. El libro combina teoría con experiencias prácticas extraídas de proyectos reales, integrando metodologías como ICONIX y RUP para ofrecer un enfoque integral.

Este libro pretende ser una guía esencial, dirigida a: analistas, desarrolladores y equipos de trabajo; ofreciendo una visión integral y práctica completa para construir productos de software que satisfagan las necesidades del cliente.

9 789942 671127