Recolección y documentación de requisitos de software

Breve descripción:

Este componente aborda la recolección y documentación de requisitos de software, destacando técnicas para identificar y registrar necesidades del cliente de manera efectiva. Incluye métodos para especificar, priorizar y mantener la trazabilidad de los requisitos, así como herramientas de control de versiones. Se enfoca en el análisis, modelado y validación de requisitos para asegurar un diseño adecuado y funcional.

Diciembre 2024

Tabla de contenido

Introducción

La recolección y documentación de requisitos de software son fundamentales para el éxito de cualquier proyecto de desarrollo. En un entorno donde las expectativas de los clientes son cada vez más altas, es indispensable aplicar técnicas y metodologías que garanticen la precisión y claridad en la definición de necesidades. Este componente proporciona un marco estructurado para identificar, recolectar y documentar requisitos, facilitando la alineación entre los objetivos del proyecto y las expectativas del cliente.

El primer aspecto relevante es la recolección de datos, donde se utilizan técnicas como entrevistas, encuestas y análisis de documentos para obtener información representativa y relevante. La identificación de fuentes de información y la creación de instrumentos de recolección son pasos requeridos para asegurar que los datos capturados reflejen las necesidades reales de los usuarios y stakeholders involucrados.

A continuación, se profundiza en la documentación de requisitos, destacando la importancia de especificar con claridad tanto los requisitos funcionales como los no funcionales. Se incluyen plantillas y estándares que aseguran una presentación consistente, así como técnicas de priorización y trazabilidad para gestionar el impacto de los cambios. Las herramientas de control de versiones resultan ser relevantes en el mantenimiento de la integridad de los documentos a lo largo del ciclo de vida del proyecto.

El análisis y modelado de requisitos es otra sección de este componente, donde se aplican técnicas de análisis para entender las necesidades del cliente y se desarrollan

diagramas y prototipos que visualizan las soluciones propuestas. Estas representaciones permiten a los equipos validar conceptos antes de la implementación, asegurando una comprensión compartida del proyecto.

Finalmente, la validación de requisitos con los clientes se aborda mediante técnicas estructuradas, garantizando que las especificaciones sean correctas y completas. Este proceso incluye simulaciones y la aplicación de protocolos para hacer ajustes según sea necesario, minimizando el riesgo de errores en las etapas posteriores.

¡Bienvenido a este recorrido por la recolección y documentación de requisitos, donde se adentrará en las mejores prácticas para desarrollar software alineado con las expectativas del cliente!

Video 1. Recolección y documentación de requisitos de software

Enlace de reproducción del video

Síntesis del video: Recolección y documentación de requisitos de software
En el componente formativo «Recolección y documentación de requisitos de software» se exploran las técnicas y metodologías necesarias para capturar y estructurar las necesidades del cliente de manera eficiente y efectiva. Durante el desarrollo de este componente, el aprendiz se familiarizará con técnicas de recolección de datos, como entrevistas, encuestas, observación y análisis de documentos, esenciales para obtener información precisa y completa. Se enseñará a identificar y seleccionar fuentes de información adecuadas, tanto primarias como secundarias, asegurando la fiabilidad y relevancia de los datos recopilados para el proyecto de software. El componente profundiza en la elaboración de instrumentos de recolección, para saber cómo diseñar cuestionarios y guías de entrevistas que optimicen la calidad de la información obtenida. La aplicación de métodos de recolección se abordará en contextos prácticos, mostrando cómo ajustar las técnicas a diferentes escenarios y necesidades del cliente. En la parte de documentación de requisitos, se desarrollará la habilidad para especificar requisitos funcionales y no funcionales, utilizando plantillas y estándares reconocidos, como IEEE 830, para asegurar claridad y consistencia.

Se abordarán herramientas de control de versiones para gestionar cambios en la documentación de requisitos, facilitando la organización y trazabilidad en equipos de desarrollo colaborativo.

La priorización y trazabilidad de los requisitos serán tratadas en detalle, explicando cómo asegurar que se dé prioridad a las funcionalidades más críticas y cómo mantener un seguimiento claro de cada requisito a lo largo del ciclo de vida del proyecto.

A lo largo del componente, se proporcionarán ejemplos y ejercicios para que el aprendiz aplique los conceptos en situaciones reales, mejorando sus competencias en el análisis y documentación de requisitos de software.

¡Bienvenido al apasionante mundo de la recolección y documentación de requisitos de software, proceso fundamental para el éxito de cualquier proyecto tecnológico!

1. Recolección de datos

La recolección de datos en el ámbito de desarrollo de software es un proceso metódico que se utiliza para captar información valiosa que servirá de base para documentar los requisitos de un sistema. Este paso inicial sirve para asegurar que las soluciones desarrolladas cumplan con las expectativas y necesidades de los usuarios y otros actores involucrados en el proyecto. A través de diversas técnicas de recolección, se busca obtener una visión integral y precisa del problema que se desea resolver y las funcionalidades que el software debe incluir.

1.1. Técnicas de recolección

La selección de técnicas de recolección depende del tipo de información que se necesita y de la disponibilidad de los usuarios y otros actores relevantes. A continuación, se detallan algunas de las técnicas más comunes y ejemplos de cómo aplicarlas:

a) **Entrevistas**: una técnica que permite obtener información cualitativa detallada directamente de los usuarios, clientes o expertos. Las entrevistas pueden ser estructuradas, semiestructuradas o abiertas.

- **Ejemplo**: si se está desarrollando un sistema de gestión para una clínica, se puede entrevistar al personal médico para entender cómo registran la información de los pacientes y qué características les gustaría que el nuevo sistema tenga.
- **Consejo**: preparar preguntas claras y abiertas, como “¿Cuáles son los desafíos más frecuentes que enfrenta con el sistema actual?” o “¿Qué características le gustaría que se mantuvieran o mejoraran?”

b) **Encuestas y cuestionarios**: herramientas útiles para recopilar datos de un gran número de personas de forma rápida. Las preguntas pueden ser cerradas (respuestas específicas) o abiertas (respuestas libres).

- **Ejemplo**: en un proyecto para un sistema de reservas de vuelos, se podría enviar un cuestionario a clientes frecuentes preguntando cómo califican su experiencia actual y qué aspectos creen que deberían mejorarse.
- **Consejo**: diseñar las preguntas para que sean fáciles de entender y no generen ambigüedades, como “¿Con qué frecuencia tiene problemas al realizar una reserva en línea?”, con opciones como "Siempre", "A menudo", "Rara vez", "Nunca".

c) **Observación**: consiste en observar cómo los usuarios interactúan con un sistema existente o cómo ejecutan sus tareas sin un sistema automatizado. La observación puede ser pasiva (sin intervenir) o participante (interactuando con los usuarios).

- **Ejemplo**: en el desarrollo de un software para una fábrica, se puede observar cómo los operarios usan una máquina y qué procesos realizan manualmente que podrían automatizarse con el nuevo sistema.
- **Consejo**: tomar notas detalladas y, si es posible, grabar las sesiones para analizarlas posteriormente.

d) **Focus group**: una reunión con un grupo de usuarios o expertos para discutir y obtener múltiples perspectivas sobre el sistema a desarrollar. Se puede usar para obtener insights sobre preferencias y prioridades.

- **Ejemplo**: si se está desarrollando una aplicación móvil para educación, se podría organizar un focus group con profesores y aprendices para entender mejor sus expectativas y desafíos.
- **Consejo**: moderar la sesión para que todos los participantes tengan la oportunidad de compartir sus opiniones, y registrar las discusiones de manera estructurada.

1.2. Identificación de fuentes de información

La identificación de fuentes de información es un paso crítico para garantizar que los datos recopilados sean precisos y relevantes para el proyecto. Estas fuentes pueden ser:

a) **Primarias**: datos obtenidos directamente de los usuarios o actores clave.
   - **Ejemplo**: si se está creando un sistema de inventario para un supermercado, las fuentes primarias podrían ser los empleados de la bodega, los gerentes de las tiendas y los proveedores.

b) **Secundarias**: datos recopilados de documentos existentes, como informes de sistemas antiguos, manuales de usuario y estudios previos.
   - **Ejemplo**: para un proyecto de rediseño de un portal web, se podría revisar la documentación previa que explique las decisiones de diseño y los informes de usabilidad.

c) **Actores involucrados**: Las personas que interactúan con el sistema o tienen un interés en su funcionamiento. Estos pueden incluir:
   - **Usuarios finales**: quienes usarán el software directamente (por ejemplo, cajeros en un sistema de punto de venta).

- **Stakeholders**: personas que tienen un interés en el éxito del sistema, como gerentes de proyecto, patrocinadores y equipos de soporte técnico.

1.3. Elaboración de instrumentos de recolección

El diseño adecuado de instrumentos de recolección de datos se utiliza para captar información útil y bien organizada. Algunos ejemplos de estos instrumentos incluyen:

a) **Guías de entrevistas**: documentos que contienen las preguntas que se realizarán durante una entrevista. Estas preguntas deben estar organizadas lógicamente y cubrir todos los aspectos importantes del sistema.
   - **Ejemplo**: para entrevistar a un gerente de ventas, una guía podría incluir preguntas como "¿Qué procesos considera más críticos en la gestión de clientes?" y "¿Cuáles son los datos más importantes que debe capturar el sistema?"

b) **Formularios de encuestas**: diseñados para obtener respuestas específicas que puedan ser analizadas cuantitativamente.
   - **Ejemplo**: un formulario de encuesta para clientes podría incluir preguntas sobre la frecuencia de uso del sistema y la satisfacción general con el servicio.

c) **Listas de verificación**: utilizadas para asegurarse de que se recopilen todos los datos necesarios.
   - **Ejemplo**: una lista de verificación para observar el uso de un sistema podría incluir ítems como "¿El usuario tiene dificultades para navegar por el menú principal?" y "¿Cuánto tiempo tarda en completar una tarea básica?"

1.4. Aplicación de métodos de recolección

La aplicación de las técnicas de recolección debe ser metódica y organizada para garantizar la validez de los datos obtenidos. Este proceso incluye:

a) **Planificación previa**: antes de iniciar la recolección, se deben definir los objetivos, elegir los métodos adecuados y asignar roles dentro del equipo.
   - **Ejemplo**: si se van a realizar observaciones en una fábrica, se debe coordinar con el personal y definir horarios que no interfieran con su productividad.

b) **Ejecución**: implementar las técnicas elegidas de manera cuidadosa, asegurando que los datos se capturen de manera estructurada.
   - **Ejemplo**: Durante una entrevista, se deben tomar notas detalladas o grabar la conversación (con el permiso del entrevistado) para no perder información importante.

c) **Documentación**: registrar los datos de manera organizada y clara para que puedan ser fácilmente analizados y utilizados en la siguiente fase del desarrollo.
   - **Ejemplo**: crear un resumen de las observaciones realizadas en una sesión de focus group, destacando los puntos importantes y las ideas más relevantes.

2. Documentación de requisitos

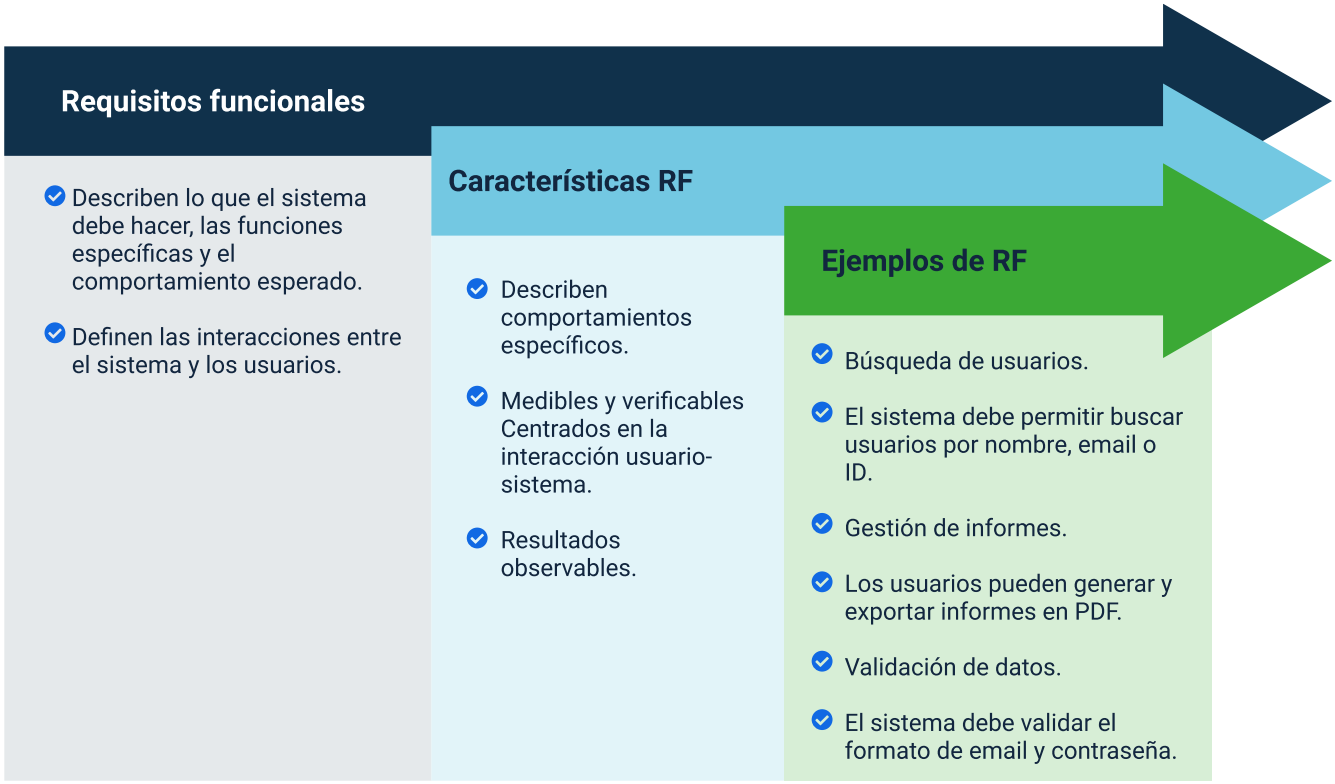
La documentación de requisitos es fundamental en el desarrollo de software, ya que establece un entendimiento claro y compartido de las expectativas del cliente y las funcionalidades necesarias. Este proceso no solo reduce la ambigüedad, sino que también facilita la evaluación y validación de los resultados en cada etapa del proyecto. En esta sección, se describen las mejores prácticas para documentar requisitos de manera estructurada, con ejemplos y técnicas que optimizan su gestión.

2.1. Especificación de requisitos funcionales y no funcionales

Los requisitos funcionales detallan las acciones que el sistema debe realizar, mientras que los no funcionales se centran en la calidad y las restricciones del sistema. Ambos deben ser definidos con precisión para evitar malentendidos durante el desarrollo.

Requisitos funcionales: incluyen las operaciones del sistema, como el inicio de sesión de usuarios, la gestión de datos, o las interacciones del usuario con el sistema.

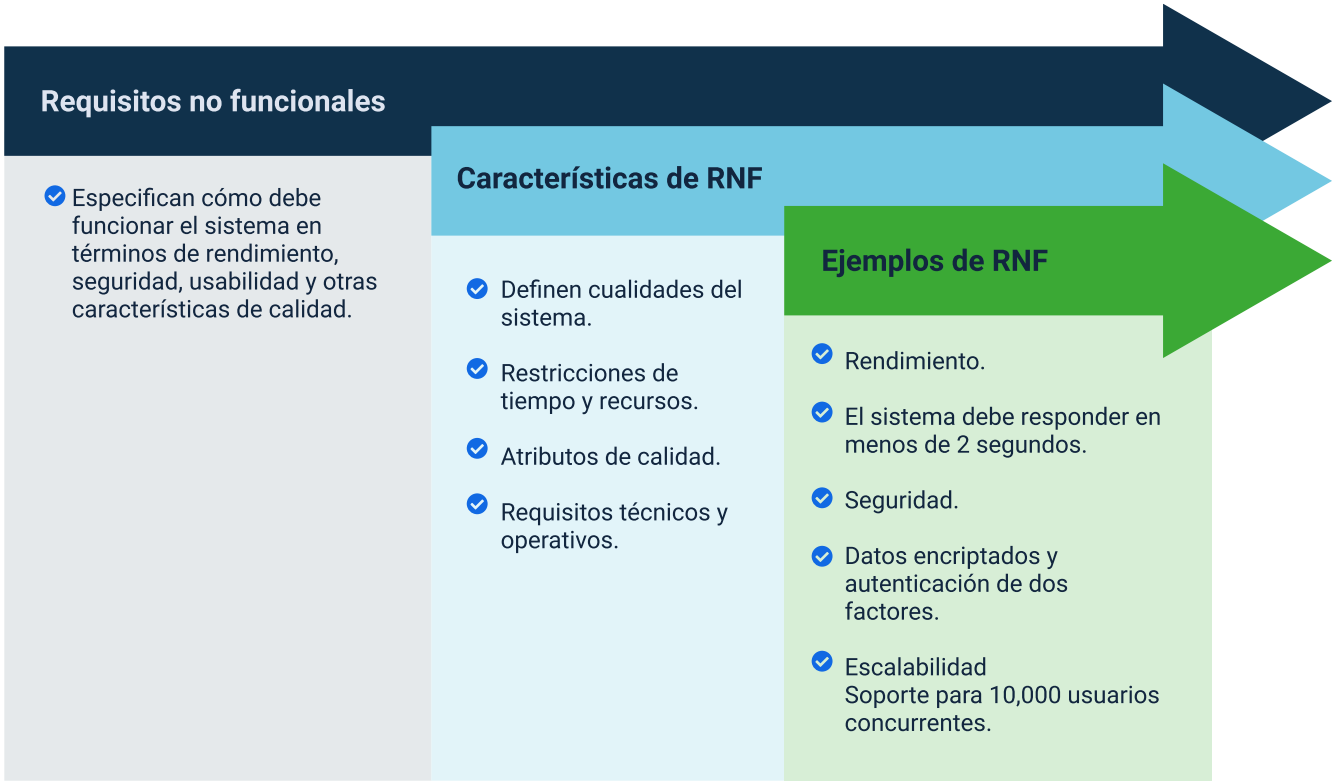
Figura 1. Requisitos funcionales



Fuente. OIT, 2024.

Requisitos no funcionales: se refieren a características como la escalabilidad, la seguridad, o la facilidad de uso.

Figura 2. Requisitos no funcionales



Fuente. OIT, 2024.

2.2. Uso de plantillas y estándares

La estandarización de la documentación a través de plantillas y normas reconocidas facilita la coherencia y la revisión de los requisitos. Las plantillas proporcionan un marco que asegura que toda la información relevante esté incluida y organizada de manera lógica.

Normas como IEEE 830 y IEEE 29148 definen estructuras para la documentación de requisitos, incluyendo secciones para el contexto del sistema, descripción detallada, restricciones, y requisitos específicos.

Tabla 1. Estándares IEEE 830 y IEEE 29148

Norma	Plantilla	Descripción	Componentes principales
IEEE 830.	Especificación de Requisitos de software (SRS).	Esta norma proporciona una estructura estándar y detallada para documentar los requisitos de software, lo que asegura una comunicación clara y consistente entre los equipos de desarrollo, diseño y los stakeholders. Permite identificar de manera precisa las necesidades funcionales y no funcionales del sistema, minimizando ambigüedades y facilitando el seguimiento del desarrollo.	• **Propósito del sistema**: describe el objetivo general y el alcance del software, así como los beneficios que proporciona. • **Descripción general**: ofrece un resumen del sistema, definiendo las principales funcionalidades y restricciones. • **Requisitos funcionales**: lista específica de funcionalidades que el software debe realizar, detallando cada operación y cómo debe comportarse. • **Requisitos no funcionales**: define atributos como seguridad, rendimiento, disponibilidad y otros aspectos de calidad. • **Interfaces externas**: explica las interacciones con otros sistemas, hardware, software y usuarios, especificando las interfaces de usuario y las conexiones necesarias.

Norma	Plantilla	Descripción	Componentes principales
IEEE 29148.	Gestión de Requisitos de Ingeniería de Sistemas.	Esta norma establece un marco integral para gestionar y especificar los requisitos de sistemas complejos. Proporciona guías para la identificación, documentación, análisis y control de los requisitos, abarcando tanto software como hardware. Facilita la trazabilidad de los requisitos desde la concepción hasta la implementación, lo cual es fundamental en proyectos de gran escala.	• **Propósito y alcance del sistema**: define claramente la finalidad del sistema y los límites de su aplicación, identificando el entorno operativo y las necesidades a satisfacer. • **Desglose de requisitos**: se detallan los requisitos de manera estructurada, especificando tanto las funciones esenciales como las características de calidad. • **Atributos de calidad de los requisitos**: evalúa y asegura que los requisitos sean completos, consistentes, viables, verificables y trazables. • **Trazabilidad y priorización**: explica cómo se relacionan los requisitos con otros elementos del proyecto y establece su importancia relativa para orientar las decisiones de desarrollo.

Fuente. OIT, 2024.

2.3. Priorización y trazabilidad

La priorización de requisitos es crítica para asegurar que el proyecto se enfoque primero en las funcionalidades de mayor valor para el cliente o el negocio. Técnicas como MoSCoW (Must, Should, Could, Won't) ayudan a categorizar los requisitos según su importancia.

Métodos de priorización:

- **MoSCoW**: asigna prioridades claras a cada requisito.
- **Análisis de valor**: Evalúa el beneficio que aporta cada funcionalidad en relación al esfuerzo necesario para implementarla.

Ejemplo: “la funcionalidad de búsqueda avanzada” podría clasificarse como una prioridad media, mientras que “La autenticación de usuarios” sería una prioridad alta.

Trazabilidad: permite rastrear cada requisito desde su concepción hasta su implementación, asegurando que ningún aspecto se pierda en el proceso. Las herramientas de trazabilidad incluyen matrices que relacionan los requisitos con las partes del sistema que los implementan, así como con las pruebas que validan su correcto funcionamiento.

Ejemplo: un requisito funcional de “autenticación de usuarios” se puede rastrear hasta el módulo de seguridad del sistema y vincularse con pruebas de validación para garantizar su correcto desempeño.

2.4. Herramientas de control de versiones

El control de versiones de los requisitos sirve para gestionar cambios, colaboraciones y mantener un historial claro de las modificaciones realizadas a lo largo del desarrollo del proyecto. Las herramientas de control de versiones proporcionan funcionalidades para revertir cambios, gestionar ramas de desarrollo, y colaborar de manera eficiente.

a) **Ejemplos de herramientas populares:**

Documentos que contienen las preguntas que se realizarán durante una entrevista. Estas preguntas deben estar organizadas lógicamente y cubrir todos los aspectos importantes del sistema.

- **Git**: ofrece control de versiones distribuido, ideal para proyectos colaborativos.

- **Subversion (SVN)**: herramienta de control de versiones centralizado, útil para proyectos que prefieren una estructura de control más tradicional.

b) Escenarios de uso:

Diseñados para obtener respuestas específicas que puedan ser analizadas cuantitativamente.

- **Gestión de cambios**: si el cliente solicita una modificación en un requisito, como ajustar la interfaz para cumplir con nuevos estándares de accesibilidad, el control de versiones permite documentar y gestionar este cambio sin perder el rastro de las versiones anteriores.
- **Colaboración**: en un equipo de desarrollo, varios miembros pueden trabajar simultáneamente en diferentes partes del documento de requisitos, con la garantía de que las versiones se gestionan y sincronizan correctamente.

Ejemplo: un cambio en la especificación de un requisito se documenta como un "commit" en la herramienta de control de versiones, permitiendo al equipo revisar, aprobar y realizar un seguimiento de las modificaciones.

3. Análisis y modelado

El análisis y modelado de requisitos es una fase del desarrollo de software. Permite transformar las necesidades del cliente en especificaciones detalladas que guíen el diseño y la implementación del sistema. Este tema se centra en técnicas y herramientas que aseguran que los requisitos se comprendan y se representen adecuadamente para todos los involucrados en el proyecto.

3.1. Análisis de necesidades del cliente

El análisis de necesidades implica identificar y entender las expectativas del cliente, así como cualquier limitación o requisito específico que pueda impactar el diseño del sistema.

a) **Métodos de análisis**: reuniones con stakeholders, entrevistas en profundidad, y observación de procesos.
   - **Ejemplo**: en un proyecto de desarrollo de una aplicación de gestión de inventario, se realizarían entrevistas con el equipo de almacén para comprender sus flujos de trabajo y necesidades específicas.

b) **Identificación de objetivos**: a partir de las necesidades recopiladas, se deben definir objetivos claros y alcanzables.
   - **Ejemplo**: si un cliente necesita reducir los tiempos de búsqueda de productos en el inventario, el objetivo sería desarrollar una función de búsqueda optimizada.

c) **Documentación de hallazgos**: los hallazgos del análisis deben documentarse de manera estructurada para facilitar su validación y seguimiento.

3.2. Creación de diagramas y prototipos

Los diagramas y prototipos son herramientas visuales que ayudan a representar la estructura y funcionalidad del sistema, facilitando la comunicación entre los desarrolladores, diseñadores y stakeholders.

Tabla 2. Diagramas y prototipos

Categoría	Subcategoría	Descripción	Ejemplo
Tipo de Diagrama.	Diagramas de flujo.	Representan el flujo lógico de las operaciones.	Un diagrama de flujo que muestra cómo un usuario navega por las opciones de un menú en una aplicación.
Tipo de Diagrama.	Diagramas de casos de uso.	Describen cómo los usuarios interactúan con el sistema.	Un caso de uso que detalla cómo un administrador agrega nuevos usuarios a la base de datos.
Prototipado.	Baja o alta fidelidad.	Crear prototipos permite obtener retroalimentación temprana de los usuarios.	Un prototipo de una aplicación de comercio electrónico que muestra el flujo de compra completo.
Prototipado.	Herramientas de prototipado.	Herramientas utilizadas para crear prototipos, desde wireframes hasta interfaces interactivas.	Figma, Adobe XD, Sketch.

Fuente. OIT, 2024.

3.3. Técnicas de modelado de requisitos

El modelado de requisitos transforma las descripciones textuales en representaciones más estructuradas, como diagramas y especificaciones formales.

a) **Lenguajes de modelado**: UML (Unified Modeling Language) es uno de los más utilizados, proporcionando una serie de diagramas que describen diferentes aspectos del sistema.

- **Diagrama de clases**: muestra la estructura de las clases, sus atributos y métodos.

  Ejemplo: un diagrama de clases para una aplicación bancaria podría incluir clases como “Cuenta”, “Cliente” y “Transacción”.

- **Diagrama de secuencia**: representa la interacción entre objetos a lo largo del tiempo.

  Ejemplo: un diagrama de secuencia que muestra cómo un usuario inicia sesión y accede a sus datos personales.

b) **Técnicas de modelado ágil**: incluyen el uso de historias de usuario y diagramas de actividades para proyectos que adoptan metodologías ágiles.

- **Historia de usuario**: “Como usuario, quiero recibir notificaciones de actualizaciones de mi pedido, para estar informado sobre el estado de mi compra.”

3.4. Integración de puntos de vista en el diseño

Es fundamental integrar los diferentes puntos de vista de los stakeholders en el diseño del sistema para garantizar que se cumplan todas las expectativas y se aborden las posibles limitaciones.

a) **Identificación de puntos de vista**: diferentes stakeholders, como usuarios finales, desarrolladores, y gerentes de proyecto, pueden tener perspectivas únicas sobre el sistema.

- **Ejemplo**: un usuario final podría priorizar la facilidad de uso, mientras que el gerente de TI podría centrarse en la seguridad de la información.

b) **Documentación de puntos de vista**: los puntos de vista se documentan utilizando herramientas como Viewpoints, que organizan las distintas perspectivas en un formato comprensible.

- **Ejemplo práctico**: en un proyecto de desarrollo de software para una clínica, los puntos de vista incluirían las necesidades de los médicos, el personal administrativo y los pacientes.

c) **Resolución de conflictos**: integrar y priorizar puntos de vista a menudo requiere negociación y compromiso, asegurando que se encuentren soluciones equilibradas.

- **Ejemplo**: si un usuario desea una interfaz muy visual y atractiva, pero el equipo de desarrollo sugiere simplicidad para mejorar el rendimiento, se podría optar por un diseño que equilibre ambos aspectos.

4. Validación de requisitos

La validación de requisitos es un proceso crítico para garantizar que los requisitos recopilados y documentados sean precisos, comprensibles y viables para los desarrolladores y el cliente. Esta fase implica verificar que los requisitos reflejen fielmente las expectativas del cliente y asegurar que el producto final cumpla con las necesidades especificadas.

4.1. Técnicas de validación con clientes

Las técnicas de validación permiten confirmar que los requisitos cumplen con las expectativas del cliente antes de proceder al desarrollo.

a) **Revisiones y walkthroughs**: consisten en reuniones donde se revisan los requisitos de manera detallada con el cliente y otros stakeholders.
   - **Ejemplo**: presentar un documento de requisitos funcionales a un grupo de stakeholders y discutir cada elemento para asegurar su comprensión y aprobación.

b) **Prototipos de alta fidelidad**: mostrar prototipos detallados a los usuarios finales para obtener su retroalimentación antes de pasar a la fase de desarrollo.
   - **Ejemplo**: un prototipo de una aplicación de reserva de vuelos donde los usuarios pueden probar la interfaz y proporcionar comentarios sobre su facilidad de uso.

c) **Simulaciones y escenarios de uso**: usar simulaciones para mostrar cómo el sistema respondería a escenarios reales.
   - **Ejemplo**: simular el flujo de una transacción bancaria en línea y pedir a los usuarios que evalúen su experiencia y satisfacción.

4.2. Revisión y ajustes de especificaciones

Después de la validación inicial, es común realizar revisiones para ajustar y mejorar los requisitos con base en la retroalimentación obtenida.

a) **Evaluación de consistencia**: revisar los requisitos para asegurarse de que no haya contradicciones o inconsistencias.
   - **Ejemplo**: verificar que un requisito de seguridad no contradiga un requisito de facilidad de acceso.

b) **Evaluación de viabilidad**: confirmar que los requisitos son factibles de implementar dentro de las limitaciones de tiempo, presupuesto y tecnología.
   - **Ejemplo**: asegurarse de que una función avanzada, como la inteligencia artificial para recomendaciones de productos, sea viable con el presupuesto y los recursos actuales.

c) **Documentación de cambios**: registrar los ajustes realizados a los requisitos y las razones detrás de estos cambios.
   - **Ejemplo**: si un cliente decide cambiar un requisito de diseño tras una sesión de validación, se documenta el cambio y se justifica en el informe de requisitos.

4.3. Simulación de validaciones con casos de estudio

El uso de casos de estudio permite simular cómo funcionaría el sistema en situaciones de la vida real, lo que ayuda a identificar problemas antes del desarrollo.

a) **Desarrollo de escenarios de prueba**: crear escenarios realistas que representen cómo los usuarios interactuarán con el sistema.

- **Ejemplo**: un caso de estudio para un sistema de gestión de inventario podría incluir la simulación de pedidos de productos, devoluciones y actualizaciones en tiempo real.

b) **Análisis de resultados**: evaluar cómo el sistema responde a las simulaciones y detectar posibles fallos o áreas de mejora.

- **Ejemplo**: si el sistema no actualiza los datos de inventario en tiempo real durante la simulación, esto indicaría un área que requiere optimización.

c) **Retroalimentación basada en simulaciones**: presentar los resultados de las simulaciones a los stakeholders y usar su retroalimentación para hacer ajustes.

- **Ejemplo**: los usuarios podrían señalar que el sistema necesita una interfaz más intuitiva después de observar cómo se comporta durante una simulación de uso.

4.4. Aplicación de protocolos aprobados

La aplicación de protocolos estándar para la validación de requisitos ayuda a garantizar la uniformidad y calidad en el proceso.

a) **Establecimiento de protocolos de validación**: crear un conjunto de reglas y procedimientos que deben seguirse para validar los requisitos.

- **Ejemplo**: un protocolo podría especificar que todos los requisitos deben ser revisados por al menos dos stakeholders antes de ser aprobados.

b) **Uso de herramientas de gestión de requisitos**: implementar software que facilite la validación y seguimiento de requisitos, como Jira, Confluence o IBM DOORS.

- **Ejemplo**: utilizar Jira para rastrear el estado de cada requisito y asegurarse de que se cumplan los protocolos de validación.

c) **Cumplimiento con estándares organizacionales**: asegurar que el proceso de validación cumpla con los estándares de calidad establecidos por la organización.

- **Ejemplo**: seguir las guías de validación definidas por el estándar IEEE 29148 para la especificación de requisitos de software.

Síntesis

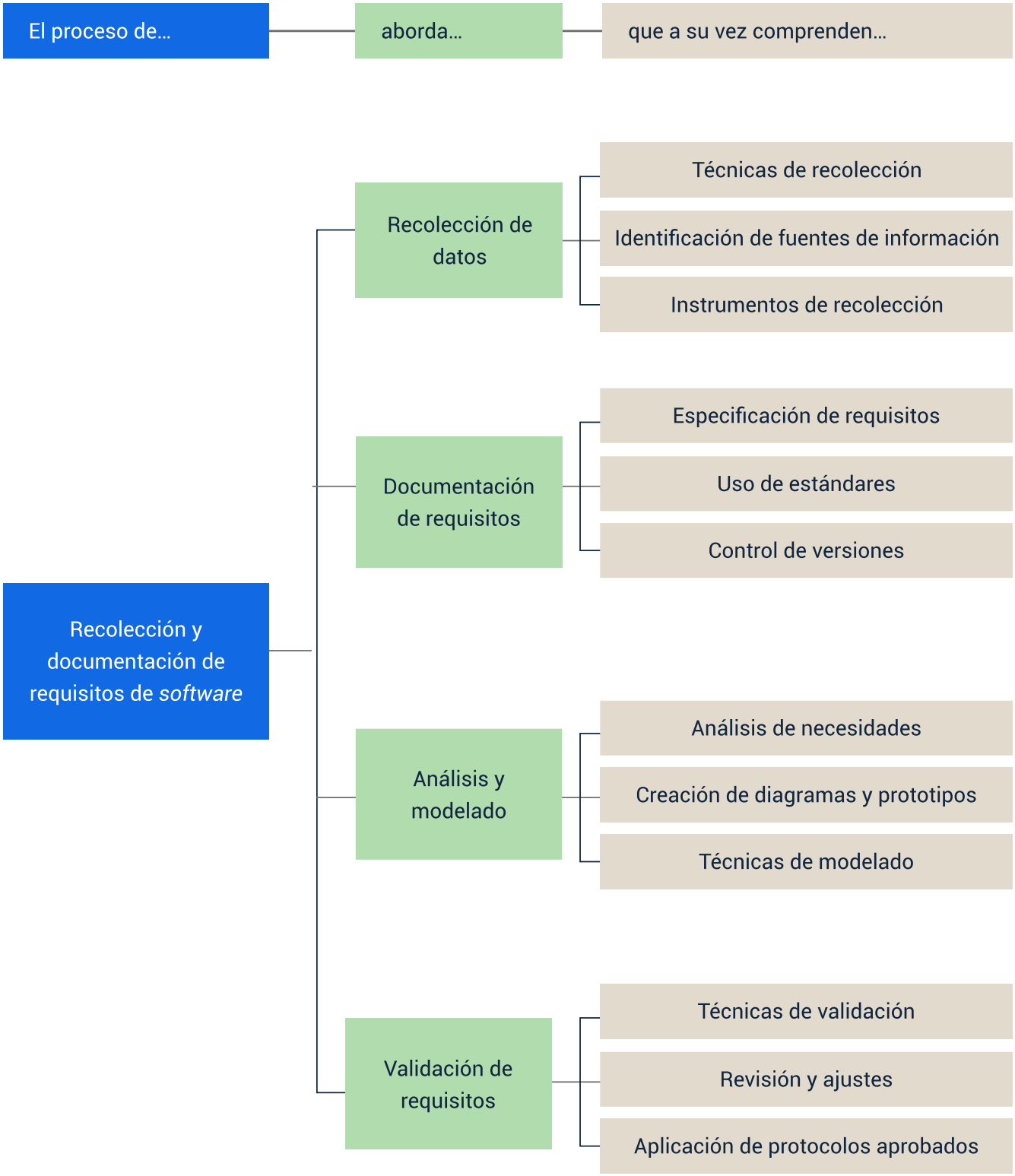
El siguiente diagrama ofrece una visión sintetizada de los principales conceptos y procedimientos desarrollados en el componente “Recolección y documentación de requisitos de software”. Está diseñado para facilitar al aprendiz la comprensión de las interrelaciones entre los diversos procesos que conforman la gestión de requisitos en un proyecto de software.

En el centro del diagrama se encuentra el concepto principal: “Recolección y documentación de requisitos de software”, del cual se desprenden cuatro procesos: recolección de datos, documentación de requisitos, análisis y modelado, y validación de requisitos. Cada uno de estos procesos se desglosa en temáticas específicas que detallan el enfoque y contenido del componente, desde técnicas de recolección hasta la validación final con el cliente.

El proceso de recolección de datos abarca la aplicación de técnicas y la elaboración de instrumentos, destacando la importancia de las fuentes de información precisas. La documentación de requisitos incluye la especificación detallada, el uso de plantillas y la gestión de versiones para mantener la integridad de los datos. En análisis y modelado, se desarrollan diagramas y prototipos que facilitan la representación visual de los requisitos. Finalmente, la validación de requisitos se centra en técnicas para asegurar que las especificaciones sean acordes a las necesidades del cliente, incluyendo revisiones y simulaciones.

Este diagrama actúa como una herramienta visual que ayuda a los aprendices a comprender la secuencia y conexión de los procesos descritos. Se les invita a utilizarlo como complemento del contenido detallado, sirviendo tanto como una referencia

rápida como un recordatorio estructurado de los elementos en la gestión de requisitos de software.


El proceso de...
aborda...
que a su vez comprenden...
Recolección y documentación de requisitos de *software*
Recolección de datos
Técnicas de recolección
Identificación de fuentes de información
Instrumentos de recolección
Documentación de requisitos
Especificación de requisitos
Uso de estándares
Control de versiones
Análisis y modelado
Análisis de necesidades
Creación de diagramas y prototipos
Técnicas de modelado
Validación de requisitos
Técnicas de validación
Revisión y ajustes
Aplicación de protocolos aprobados

Material complementario

Tema	Referencia	Tipo de material	Enlace del recurso
1. Recolección de datos	Ecosistema de Recursos Educativos Digitales SENA. (2023e, septiembre 7). La fase de elicitación de requisitos.	Video	https://www.youtube.com/watch?v=-9NsuoSa_Ao
1. Recolección de datos	Ecosistema de Recursos Educativos Digitales SENA. (2021, 26 noviembre). Diagramas para la especificación y análisis de requisitos: introducción.	Video	https://www.youtube.com/watch?v=N1zFo2-dmkU
2. Documentación de requisitos	Ecosistema de Recursos Educativos Digitales SENA. (2023c, mayo 26). Requerimientos y procesos del sistema de información.	Video	https://www.youtube.com/watch?v=7LYV0UoH82o
2. Documentación de requisitos	Ecosistema de Recursos Educativos Digitales SENA. (2023a, marzo 25). Construcción del informe de requisitos.	Video	https://www.youtube.com/watch?v=F_sGSI26q88
3. Análisis y modelado	Ecosistema de Recursos Educativos Digitales SENA. (2023d, septiembre 7). Análisis y especificación de requisitos.	Video	https://www.youtube.com/watch?v=Hmtriz7Af20
3. Análisis y modelado	Ecosistema de Recursos Educativos Digitales SENA. (2022a, diciembre 13). Análisis de requisitos, procesos e información.	Video	https://www.youtube.com/watch?v=iSWz9b7HCEo

Tema	Referencia	Tipo de material	Enlace del recurso
4. Validación de requisitos	Ecosistema de Recursos Educativos Digitales SENA. (2023b, marzo 27). Validación del informe de los requisitos según las directrices de negocio y solicitud del cliente.	Video	https://www.youtube.com/watch?v=6l13UShHdP8
4. Validación de requisitos	Ecosistema de Recursos Educativos Digitales SENA. (2022b, diciembre 26). Validación de requisitos.	Video	https://www.youtube.com/watch?v=AC_xUTvJ43Q

Glosario

Análisis de necesidades: evaluación detallada de los requisitos del cliente para asegurar que se entiendan completamente antes de la implementación.

Control de versiones: sistema que gestiona los cambios realizados en los documentos y mantiene un historial de las revisiones.

Diagrama: representación gráfica que ayuda a visualizar procesos, estructuras de datos o interacciones del sistema.

Fuentes de información: personas, documentos o sistemas de los cuales se extrae información útil para definir requisitos de software.

Informe de requisitos: documento que describe en detalle los requisitos funcionales y no funcionales del sistema, incluyendo análisis y especificaciones técnicas.

Instrumentos de recolección: herramientas y formatos, como cuestionarios y guías de entrevistas, que facilitan la obtención de datos de manera estructurada.

Plantillas y estándares: documentos predefinidos y normas que aseguran la uniformidad y calidad en la especificación de requisitos.

Priorización de requisitos: proceso de clasificar los requisitos según su importancia y urgencia para el proyecto.

Protocolos aprobados: conjunto de procedimientos estándar que se siguen para realizar validaciones y asegurar la calidad de los requisitos.

Prototipo: modelo preliminar de la interfaz de usuario o funcionalidad del software que se usa para obtener retroalimentación y validar requisitos.

Puntos de vista: diferentes perspectivas que se consideran en el diseño del sistema, como las necesidades del usuario, del cliente y del desarrollador.

Recolección de datos: proceso de obtener información relevante de diferentes fuentes para entender las necesidades del cliente y los requisitos del sistema.

Requisitos funcionales: describen las funciones específicas que el software debe realizar, como operaciones, cálculos o interacciones con el usuario.

Requisitos no funcionales: definen las cualidades del sistema, como rendimiento, seguridad, usabilidad y escalabilidad.

Revisión de especificaciones: evaluación de los documentos de requisitos para asegurar que sean precisos, completos y comprensibles.

Simulación: uso de modelos o escenarios para probar cómo se comportará el sistema en situaciones reales antes de su desarrollo.

Técnicas de modelado: métodos como diagramas de flujo, casos de uso o mapas de navegación, utilizados para representar requisitos de manera estructurada.

Técnicas de recolección: métodos utilizados para obtener datos, como entrevistas, encuestas, observación, y focus group.

Trazabilidad: capacidad de rastrear cada requisito a lo largo de su ciclo de vida, desde la recolección hasta su implementación y pruebas.

Validación de requisitos: proceso de confirmar que los requisitos reflejan las expectativas del cliente y son viables para su desarrollo.

Referencias bibliográficas

Ambler, S. W. (2002). The Agile Model Driven Development (AMDD) Process. https://agilemodeling.com/essays/amdd.htm

Davis, A. M. (2005). Software Requirements: Objects, Functions, and States. Prentice Hall. https://archive.org/details/softwarerequirem0000davi

Gotel, O. C. Z., & Finkelstein, A. (1994). An analysis of the requirements traceability problem. In Proceedings of the First International Conference on Requirements Engineering (pp. 94-101). IEEE. https://www.semanticscholar.org/paper/An-analysis-of-the-requirements-traceability-Gotel-Finkelstein/ebc86c81ace4607f3f59a9053ec542cf323140a2

Hoy, Z., & Xu, M. (2023, June 1). Agile Software Requirements Engineering Challenges-Solutions—A Conceptual Framework from Systematic Literature Review. Information (Switzerland). MDPI. https://doi.org/10.3390/info14060322

IEEE. (1998). IEEE Std 830-1998: IEEE Recommended Practice for Software Requirements Specifications. https://www.math.uaa.alaska.edu/~afkjm/cs401/IEEE830.pdf

Karlsson, J., & Ryan, K. (1997). A cost-value approach for prioritizing requirements. In Proceedings of the 2nd International Conference on Requirements Engineering (p. 97-104). IEEE. https://www.semanticscholar.org/paper/A-Cost-Value-Approach-for-Prioritizing-Requirements-Karlsson-Ryan/a28fdbaf1885dee544a1d4899d90994632f47564

Leffingwell, D., & Widrig, D. (2003). Managing Software Requirements: A Unified Approach. Addison-Wesley. https://books.google.ca/books/about/Managing_Software_Requirements.html?id=h4pPpXp-xrEC

Letaw, L. (2024, 1 enero). Requirements. Handbook Of Software Engineering Methods. https://open.oregonstate.education/setextbook/chapter/requirements/

Mantilla, G. (s. f.). Ingeniería de Requisitos acorde a K. Pohl. https://raven4j.blogspot.com/2015/08/ingenieria-de-requisitos-acorde-k-pohl.html

Martin, M. (2022). What is a Functional Requirement in Software Engineering? Specification, Types, Examples. Guru99. https://www.guru99.com/functional-requirement-specification-example.html

Melegati, J., Goldman, A., Kon, F., & Wang, X. (2019). A model of requirements engineering in software startups. Information and Software Technology, 109, 92–107. https://doi.org/10.1016/j.infsof.2019.02.001

Serna M., E. (2021). Métodos Formales, Ingeniería de Requisitos y Pruebas del Software. Zenodo. https://doi.org/10.5281/zenodo.4534359

Sommerville, I. (2016). Software Engineering (10th ed.). Pearson. https://www.pearson.com/en-us/subject-catalog/p/software-engineering/P200000003258/9780137503148

Créditos

Elaborado por:

Organización Internacional del Trabajo