Universidad Nacional

Autónoma de México

Facultad de ingeniería

Documentación del tutorial.

Ingeniería en software

Grupo: 4

Equipo: 5

Semestre: 2026 – 1

Profesor: Orlando Zaldívar Zamorategui

Fecha de entrega:

13 de Noviembre 2025

Líder del equipo: Domínguez Ruiz Samantha Atziry - 322273672

Temario: Fundamentos y Técnicas de Ingeniería de Requisitos de Software

Módulo 1: Introducción a los Requerimientos de Software

1.1. Importancia y Contexto

1.2. Planteamiento del Problema

Módulo 2: Fundamentos de los Requerimientos

2.1. Propósitos y Beneficios Clave

2.2. Clasificación de Requerimientos

2.2.1. Requerimientos Funcionales (RF)

2.2.2. Requerimientos No Funcionales (RNF)

Módulo 3: Técnicas de Obtención de Requerimientos

3.1. Técnicas Tradicionales

3.1.1. Entrevistas

3.1.2. Cuestionarios y Encuestas

3.1.3. Observación Directa

3.1.4. Análisis de Documentación

3.2. Técnicas Colaborativas

3.2.1. Talleres de Trabajo (Workshops)

3.2.2. Tormenta de Ideas (Brainstorming)

3.3. Técnicas Visuales e Interactivas

3.3.1. Prototipado

3.3.2. Historias de Usuario

Módulo 4: Proceso de Ingeniería de Requisitos

4.1. Fases del Proceso

Fundamento para los requerimientos del software.
Técnicas para obtener requerimientos

Objetivo de aprendizaje: Comprender la importancia de los requerimientos de software, identificar sus tipos y conocer las principales técnicas para su obtención y gestión efectiva.

Planteamiento del problema: En muchos proyectos de software, los fallos no se deben al código, sino a una mala comprensión de los requerimientos. Esta situación genera pérdidas de tiempo, sobrecostos y productos que no cumplen con las expectativas del cliente.

Los requerimientos de software son simplemente una descripción de lo que un programa de software en particular debe hacer. Actúan como pautas para que los desarrolladores creen un producto funcional que satisfaga las necesidades de los usuarios. Un ejemplo análogo es, si los cimientos de un edificio son débiles, todo el edificio fallará; lo mismo pasa con el software y sus requerimientos.

Un buen levantamiento y gestión de requerimientos es esencial para el éxito de un proyecto, ya que los errores cometidos en esta etapa inicial suelen ser los más costosos de corregir en fases posteriores. Si los requerimientos son incompletos, ambiguos o contradictorios, el software resultante probablemente no cumplirá con las expectativas de los usuarios.

Realizar una buena gestión de requerimientos es crucial para el éxito del proyecto. Por lo que los principales fundamentos de los requerimientos son:

- **Alineación:** Permiten que todas las partes interesadas (clientes, usuarios, desarrolladores y gerentes de proyecto) comprendan y compartan una visión común del sistema.

- **Guía:** Los requerimientos son la referencia principal para las etapas de diseño, implementación, prueba y mantenimiento.

- **Validación:** Sirven como criterio de evaluación para verificar si el producto final satisface las necesidades planteadas.

- **Control del alcance:** Ayudan a definir los límites del sistema, evitando agregar funcionalidades no planificadas (lo que se conoce como *scope creep*).

- **Gestión de riesgos:** Una definición clara de requerimientos reduce la posibilidad de fallos técnicos o malentendidos con el cliente.

Existen 2 tipos de requerimiento:

- **Requerimientos Funcionales (RF)**

Describen las funciones o servicios que el sistema debe proveer.
Algunos ejemplos:

- Ejemplo 1: "El usuario **debe poder** iniciar sesión con su correo y contraseña."
- Ejemplo 2: "El sistema **debe generar** un reporte de ventas mensual en formato PDF."
- Ejemplo 3: "El administrador **debe poder** agregar o eliminar usuarios del sistema."

- **Requerimientos No Funcionales (RNF)**

Describen las cualidades, atributos o restricciones del sistema. No se refieren a *qué* hace, sino a *cómo* lo hace. Son tan importantes como los funcionales.
Por ejemplo:

- Rendimiento: "La página de inicio debe cargar en menos de 2 segundos."
- Seguridad: "Todas las contraseñas de usuario deben almacenarse cifradas usando el algoritmo SHA-256."
- Usabilidad: "Un usuario nuevo debe poder completar una compra en menos de 5 clics."
- Fiabilidad: "El sistema debe tener una disponibilidad del 99.9%."
- Portabilidad: "La aplicación debe funcionar en los navegadores Chrome, Firefox y Safari."

Técnicas para obtener requerimientos

El proceso de **ingeniería de requerimientos** implica identificar, analizar, documentar, validar y gestionar las necesidades del cliente. Para ello se emplean diversas técnicas, según el tipo de proyecto, la cantidad de usuarios y el nivel de detalle requerido.

1. Entrevistas

Una de las técnicas más comunes. Se realizan reuniones entre los analistas y los stakeholders para obtener información directa.

- **Ventajas:** Permite aclarar dudas y obtener información detallada.
- **Desventajas:** Puede ser lenta si hay muchos involucrados.
- **Tipos:** Estructuradas (preguntas definidas), semiestructuradas o abiertas.

2. Cuestionarios o Encuestas

Permiten recopilar datos de muchos usuarios de forma rápida y económica.

- **Ventajas:** Alcance amplio y fácil análisis estadístico.
- **Desventajas:** Las respuestas pueden ser superficiales o poco precisas.

3. Observación Directa

El analista observa cómo los usuarios realizan sus tareas en su entorno real de trabajo.

- **Ventajas:** Permite descubrir requerimientos no expresados verbalmente.
- **Desventajas:** Puede ser intrusiva o influir en el comportamiento del usuario.

4. Análisis de Documentación Existente

Se revisan manuales, reportes, formularios o sistemas anteriores para entender los procesos actuales y las restricciones del entorno.

- **Ventajas:** Útil cuando no se dispone de mucho tiempo con los usuarios.
- **Desventajas:** La información puede estar desactualizada o incompleta.

5. Talleres de Trabajo (Workshops)

Sesiones colaborativas donde usuarios, analistas y desarrolladores discuten y priorizan requerimientos.

- **Ventajas:** Fomenta la participación activa y la alineación del equipo.
- **Desventajas:** Requiere buena organización y moderación.

6. Prototipado

Se crean versiones preliminares del sistema (mockups o prototipos interactivos) para que los usuarios visualicen cómo funcionará el software.

- **Ventajas:** Permite validar requerimientos antes del desarrollo completo.
- **Desventajas:** Puede generar expectativas irreales si no se aclara que es solo un modelo.

7. Tormenta de Ideas (Brainstorming)

Reuniones creativas en las que los participantes proponen libremente funcionalidades o mejoras posibles.

- **Ventajas:** Promueve la creatividad y la identificación de oportunidades.
- **Desventajas:** Puede generar ideas poco viables si no se filtran adecuadamente.

8. Historias de Usuario (User Stories)

Usadas especialmente en metodologías ágiles. Describen una funcionalidad desde la perspectiva del usuario final.

- **Ejemplo:** “Como comprador, quiero recibir una notificación cuando mi pedido sea enviado, para saber cuándo llegará.”
- **Ventajas:** Fáciles de entender y mantener.
- **Desventajas:** No siempre reflejan todos los detalles técnicos necesarios.

Conclusión

Los requerimientos de software son el **pilar fundamental** de todo proyecto de desarrollo. Una correcta obtención y gestión garantiza que el producto final cumpla con las expectativas del cliente, sea confiable, funcional y de calidad. Las técnicas de levantamiento deben elegirse según el tipo de proyecto, el número de usuarios y la metodología empleada (tradicional o ágil). En resumen, **sin buenos requerimientos, no puede haber buen software**.

Ejemplos

Ejemplo 1: Sistema de Gestión de Biblioteca Universitaria

1. Planteamiento del Problema

La biblioteca de la "Universidad del Valle" maneja sus préstamos y devoluciones de libros de forma manual, con registros en papel. Esto genera los siguientes problemas:

- Pérdida de registros de préstamos.

- Retrasos en la renovación de libros.

- Dificultad para consultar la disponibilidad de un libro en tiempo real.

- No hay un control eficiente de multas por retrasos.

Consecuencia: Los estudiantes y bibliotecarios están insatisfechos, y se requiere un sistema de software para automatizar los procesos.

2. Obtención de Requerimientos (Técnicas Aplicadas)

a) Entrevistas (Semiestructuradas)

- Entrevistados: Bibliotecario jefe, asistente de biblioteca, estudiantes frecuentes.

- Preguntas clave:

- ¿Qué procesos les gustaría automatizar?

- ¿Qué información debe mostrar el sistema al buscar un libro?

- ¿Cómo manejan actualmente las multas?

- Hallazgos:

- Necesidad de un catálogo en línea.

- Control automático de multas.

- Notificaciones por correo electrónico para recordatorios de devolución.

b) Observación Directa

- Actividades observadas:

- Proceso de préstamo: El bibliotecario busca en un archivo de Excel si el libro está disponible, luego llena una ficha de préstamo.

- Devolución: Verifica la fecha de devolución y calcula manualmente las multas.

- Requerimiento identificado: El sistema debe calcular automáticamente las multas según los días de retraso.

c) Prototipado

- Herramienta: Figma.

- Prototipo de pantalla de búsqueda:

- Campo de búsqueda por título, autor o ISBN.

- Resultados que muestran: título, autor, estante, disponibilidad (Sí/No).

- Feedback del usuario: Solicitaron incluir un filtro por categoría (ej.: "Ciencia Ficción").

3. Requerimientos Documentados

Requerimientos Funcionales (RF)

- RF-001: El sistema debe permitir a los usuarios buscar libros por título, autor, ISBN o categoría.

- RF-002: El sistema debe permitir el préstamo de libros solo si el usuario no tiene multas pendientes.

- RF-003: El sistema debe calcular automáticamente multas por retraso ($1 por día).

- RF-004: El sistema debe enviar un correo electrónico de recordatorio 2 días antes de la fecha de devolución.

- RF-005: El bibliotecario debe poder agregar, editar o eliminar libros del catálogo.

Requerimientos No Funcionales (RNF)

- RNF-001 (Rendimiento): La búsqueda de libros debe devolver resultados en menos de 3 segundos.

- RNF-002 (Seguridad): Las contraseñas de los usuarios deben almacenarse cifradas con bcrypt.

- RNF-003 (Usabilidad): Un usuario debe poder realizar un préstamo en menos de 4 clics.

- RNF-004 (Fiabilidad): El sistema debe estar disponible el 99.5% del tiempo en horario de biblioteca (7:00 a 22:00).

- RNF-005 (Portabilidad): El sistema debe ser accesible desde Chrome, Edge y Safari.

4. Validación y Priorización

- Taller de trabajo con stakeholders: Se priorizaron los RF-001 y RF-003 como críticos para la primera versión.

- Historias de usuario creadas:

  - Como estudiante, quiero buscar libros por autor, para encontrar rápidamente material de estudio.

  - Como bibliotecario, quiero que el sistema calcule multas automáticamente, para ahorrar tiempo y reducir errores.

Ejemplo 2: Aplicación Móvil de Delivery de Comida

1. Planteamiento del Problema

Un restaurante local "Sabor Casero" quiere expandir sus ventas mediante una aplicación móvil. Actualmente, los pedidos se reciben por teléfono, lo que genera:

- Errores en los pedidos por malentendidos.

- Falta de seguimiento al estado del pedido.

- Pérdida de clientes por no ofrecer un servicio moderno.

2. Obtención de Requerimientos (Técnicas Aplicadas)

a) Cuestionarios o Encuestas

- Preguntas para clientes frecuentes:

- ¿Qué funcionalidades esperaría en una app de delivery?

- ¿Prefiere pagar en línea o en efectivo?

- ¿Le gustaría recibir notificaciones sobre ofertas?

- Resultados: 80% prefiere pago en línea; 60% quiere seguimiento en tiempo real del pedido.

b) Tormenta de Ideas (Brainstorming)

- Participantes: Dueño del restaurante, chef, repartidor, desarrollador.

- Ideas generadas:

  - Sistema de puntos por compras recurrentes.

  - Opción de personalizar platos (ej.: sin picante).

  - Integración con Google Maps para seguimiento de repartidores.

- Filtrado: Se descartó "sistema de puntos" por complejidad en la primera versión.

c) Historias de Usuario

- Ejemplo detallado:

  - Título: Seguimiento de pedido en tiempo real.

  - Como cliente, quiero ver en un mapa la ubicación de mi repartidor, para saber cuándo llegará mi pedido.

  - Criterios de aceptación:

    1. La app debe mostrar un mapa con la ubicación en tiempo real del repartidor.

    2. El repartidor debe actualizar su estado ("En camino", "Entregado").

    3. La actualización de la ubicación debe ser cada 30 segundos.

3. Requerimientos Documentados

Requerimientos Funcionales (RF)

- RF-001: La app debe permitir al usuario registrarse con correo electrónico y contraseña.

- RF-002: La app debe mostrar un menú con categorías (ej.: "Entradas", "Platos Fuertes").

- RF-003: El usuario debe poder personalizar su plato (ej.: agregar o quitar ingredientes).

- RF-004: La app debe integrarse con un sistema de pago en línea (ej.: PayPal).

- RF-005: El usuario debe recibir notificaciones push sobre el estado de su pedido (Confirmado, En preparación, En camino, Entregado).

Requerimientos No Funcionales (RNF)

- RNF-001 (Rendimiento): La app debe cargar el menú en menos de 2 segundos.

- RNF-002 (Seguridad): Los datos de pago deben transmitirse usando TLS 1.3.

- RNF-003 (Usabilidad): Un usuario nuevo debe poder hacer un pedido en menos de 5 minutos.

- RNF-004 (Portabilidad): La app debe funcionar en Android (versión 10+) e iOS (versión 14+).

4. Validación y Control de Alcance

- Prototipado interactivo: Se usó Marvel App para simular el flujo de pedido.

- Feedback: Los usuarios sugirieron agregar un historial de pedidos.

- Control de alcance: Se pospuso "historial de pedidos" para la versión 2.0, priorizando las funcionalidades básicas.

Ejemplo 3: Sistema de Gestión de Incidencias para Soporte Técnico

1. Planteamiento del Problema

Una empresa de TI "TechSolutions" maneja incidencias de clientes mediante correos electrónicos y hojas de cálculo. Esto causa:

- Incidencias pérdidas o duplicadas.

- Falta de seguimiento al tiempo de resolución.

- Dificultad para generar reportes de rendimiento del equipo.

2. Obtención de Requerimientos (Técnicas Aplicadas)

a) Análisis de Documentación Existente

- Documentos revisados: Planillas de Excel con incidencias, correos de clientes.

- Hallazgos:

  - 40% de las incidencias son repetitivas (ej.: "Error 404 en la web").

  - No hay un registro del tiempo promedio de resolución.

b) Talleres de Trabajo (Workshops)

- Participantes: Gerente de soporte, técnicos, clientes clave.

- Actividades:

  - Lluvia de ideas para categorizar incidencias (Críticas, Altas, Medias, Bajas).

  - Definición de flujo: Recepción → Asignación → Resolución → Cierre.

- Acuerdos: Se priorizó la automatización de la asignación por área técnica.

c) Historias de Usuario

- Ejemplo:

  - Como técnico, quiero recibir notificaciones de incidencias asignadas, para resolverlas rápidamente.

  - Criterios de aceptación:

    1. El sistema debe asignar incidencias automáticamente según la especialidad del técnico (ej.: red, base de datos).

    2. El técnico debe recibir un correo con los detalles de la incidencia.

3. Requerimientos Documentados

Requerimientos Funcionales (RF)

- RF-001: El sistema debe permitir a los clientes registrar incidencias por correo electrónico o mediante un formulario web.

- RF-002: El sistema debe categorizar automáticamente las incidencias según palabras clave (ej.: "lentitud" → Baja, "caída" → Crítica).

- RF-003: El sistema debe generar reportes mensuales con métricas: tiempo promedio de resolución, incidencias por técnico.

- RF-004: El administrador debe poder asignar incidencias manualmente a técnicos específicos.

Requerimientos No Funcionales (RNF)

- RNF-001 (Fiabilidad): El sistema debe tener una disponibilidad del 99.9% en horario laboral (8:00 a 18:00).

- RNF-002 (Rendimiento): La carga de reportes debe tomar menos de 10 segundos.

- RNF-003 (Seguridad): Solo los técnicos y administradores autorizados pueden acceder a las incidencias.

4. Gestión de Riesgos

- Riesgo identificado: Los clientes podrían no adaptarse al nuevo sistema.

- Mitigación: Se incluyó un manual de usuario y capacitación en línea.

- Seguimiento: Se programaron reuniones semanales con el cliente para ajustar requerimientos.

Bibliografía

Ingeniería de Requisitos: Software Orientado al Negocio – Carlos Eduardo Vázquez y Guilherme Siqueira Simões. Español.

Fundamentos de Ingeniería de los Requisitos. Manual didáctico – Isabel María del Águila Cano. Español.

Gestión de la Ingeniería de Requisitos Integrando Principios del Pensamiento Complejo – Edgar Serna M. Español.

Requirements Engineering: Fundamentals, Principles, and Techniques – Klaus Pohl. Springer, 2010 (y ediciones posteriores).

Requirements Engineering – Elizabeth Hull, Ken Jackson, Jeremy Dick. Springer, 2017 (4ª edición).

Requirements Engineering: Laying a Firm Foundation – James A. Crowder & Curtis W. Hoff. Springer, 2022..

User-Centred Requirements Engineering – Alistair Sutcliffe. Springer, 2002.

The Requirements Engineering Handbook – Ralph R. Young. Artech House, 2004.

Requirements Engineering for Software and Systems – Phillip A. Laplante & Mohamad Kassab. Routledge, 2022 (4ª edición).

Software Requirements Essentials: Core Practices for Successful Business Analysis – Karl Wiegers & Candase Hokanson. Pearson, 2023.

Software Requirements Engineering – Richard H. Thayer & Merlin Dorfman (eds). Wiley, 1997.