Ingeniería de requisitos
© Universitas Europaea IMF

campus.euniv.eu © Universitas Europaea IMF
Juan Ulises PÉREZ VISAIRAS
campus.euniv.eu © Universitas Europaea IMF
Juan Ulises PÉREZ VISAIRAS
campus.euniv.eu © Universitas Europaea IMF
Juan Ulises PÉREZ VISAIRAS

Indice

campus.eunivers.eu © Universitas Europaea IMF Juan Ulises PÉREZ VISAIRAS

Universidad Europea IMF

Ingeniería de requisitos

1. Introducción

La ingeniería de requisitos es una disciplina clave en el desarrollo de software. Consiste en identificar, analizar, documentar y gestionar las necesidades del cliente y las restricciones del sistema, asegurando que el producto final cumpla con las expectativas y los objetivos definidos, minimizando riesgos y errores durante el desarrollo.

1.1 Definición de la ingeniería de requisitos

La ingeniería de requisitos es el proceso sistemático y estructurado de identificar, analizar, documentar, verificar y gestionar las necesidades y expectativas de los stakeholders respecto a un sistema o producto. Su objetivo es garantizar que el software desarrollado cumpla con los requisitos funcionales (qué debe hacer el sistema) y no funcionales (cómo debe comportarse), así como con las restricciones técnicas y normativas.

Este proceso incluye varias actividades clave, como la recolección de requisitos, donde se recopilan las necesidades iniciales, el análisis para detectar inconsistencias o conflictos, la especificación para documentar de manera clara y precisa los requisitos, y la validación para asegurar que estos sean correctos y estén alineados con los objetivos del proyecto.

La ingeniería de requisitos es fundamental en el ciclo de vida del software, ya que establece una base sólida para el diseño, implementación y mantenimiento del sistema, minimizando riesgos y costos asociados a cambios tardíos o errores no detectados.

1.2. Importancia de los requisitos en el ciclo de vida del software

Los requisitos son el pilar fundamental en el ciclo de vida del software, ya que establecen qué necesita y espera el cliente del sistema. Una definición clara y precisa de los requisitos asegura que todas las etapas posteriores del desarrollo, como el diseño, la implementación y las pruebas, se realicen de manera alineada con los objetivos del proyecto.

Cuando los requisitos están bien definidos, se minimizan riesgos como malentendidos, funcionalidades innecesarias o conflictos entre los stakeholders. Además, ayudan a priorizar las tareas del equipo de desarrollo, optimizando recursos y tiempo.

Por otro lado, la falta de claridad en los requisitos puede generar errores costosos en etapas avanzadas, retrasos en la entrega o productos finales que no satisfacen las necesidades del cliente. En este contexto, la validación y gestión de los requisitos a lo largo del proyecto son esenciales para garantizar la calidad y el éxito del software.

1.3. Objetivos y alcance de la ingeniería de requisitos

La ingeniería de requisitos tiene como objetivo principal garantizar que el software desarrollado cumpla con las expectativas de los stakeholders y los objetivos del proyecto. Esto se logra a través de un proceso estructurado que identifica, analiza, documenta y gestiona los requisitos funcionales y no funcionales del sistema.

Entre sus objetivos específicos se incluyen:

Documento solo para uso personal. Prohibido copiar o distribuir

1. **Definir claramente las necesidades del cliente:** Garantizar que todos los stakeholders comprendan los objetivos del proyecto y estén alineados.
2. **Identificar y resolver conflictos:** Detectar inconsistencias o discrepancias entre los requisitos y las expectativas para evitarlas en etapas avanzadas.
3. **Proporcionar una base sólida para el desarrollo:** Crear un marco que guíe las fases de diseño, implementación y pruebas.
4. **Gestionar cambios en los requisitos:** Establecer mecanismos para adaptarse a modificaciones sin comprometer la calidad ni los plazos del proyecto.

El alcance de la ingeniería de requisitos incluye todas las actividades relacionadas con el ciclo de vida del software, desde la recopilación inicial de necesidades hasta la gestión de cambios y validación del producto final. También abarca la comunicación efectiva entre los stakeholders y el equipo técnico, asegurando que el sistema desarrollado sea funcional, escalable y sostenible. Este enfoque integral minimiza riesgos y maximiza la satisfacción del cliente.

2. Tipos de Requisitos

Los requisitos son fundamentales para definir qué debe hacer un sistema y cómo debe comportarse. Se clasifican en funcionales, no funcionales, técnicos y legales, abarcando desde las funcionalidades del software hasta las características de rendimiento, restricciones tecnológicas y normativas que deben cumplirse durante el desarrollo.

2.1. Requisitos funcionales

Los requisitos funcionales describen las funcionalidades específicas que un sistema debe realizar para cumplir con las necesidades del usuario. Estos requisitos definen **qué debe hacer el software**, como procesar datos, interactuar con el usuario o integrar otros sistemas. Son fundamentales porque establecen la base de las funcionalidades principales del producto.

Definición y ejemplos

Un requisito funcional se caracteriza por ser observable y medible desde el punto de vista del usuario. Por ejemplo:

1. El sistema debe permitir a los usuarios registrarse y autenticar su cuenta.
2. Los usuarios podrán buscar productos por categoría y añadirlos a un carrito de compras.
3. El software debe generar reportes mensuales de ventas en formato PDF.
4. El sistema debe enviar notificaciones por correo electrónico cuando se complete una transacción.

Estos ejemplos destacan cómo los requisitos funcionales están directamente relacionados con las acciones esperadas de un sistema.

Identificación de casos de uso:

Los casos de uso son una herramienta clave para identificar y documentar los requisitos funcionales. Representan escenarios específicos en los que el usuario interactúa con el sistema para lograr un objetivo. Por ejemplo, un caso de uso típico sería: "El usuario inicia sesión para acceder a su perfil y consultar su historial de compras".

Cada caso de uso se compone de:

Documento solo para uso personal. Prohibido copiar o distribuir

- **Actor:** El usuario o sistema externo que interactúa con el software.
- **Flujo principal:** Los pasos secuenciales necesarios para cumplir el objetivo.
- **Excepciones:** Condiciones que podrían interrumpir el flujo principal, como errores de autenticación.

La identificación de casos de uso permite detallar los requisitos funcionales de manera estructurada, asegurando que cada funcionalidad esté alineada con las necesidades del usuario y sea verificable en las etapas de pruebas.

2.2. Requisitos no funcionales

Los requisitos no funcionales describen las características de calidad que un sistema debe cumplir para garantizar su rendimiento, seguridad, usabilidad y otras propiedades esenciales. A diferencia de los requisitos funcionales, que especifican qué debe hacer el sistema, los no funcionales definen **cómo debe comportarse** y bajo qué condiciones debe operar.

Los requisitos no funcionales son esenciales para la experiencia del usuario y el éxito del sistema, asegurando que este sea confiable, eficiente y cumpla con estándares de calidad.

Características:

1. **Rendimiento:** Define la velocidad y capacidad de respuesta del sistema. Ejemplo: "El sistema debe procesar 100 transacciones por segundo con un tiempo de respuesta inferior a 2 segundos."
2. **Seguridad:** Especifica cómo proteger los datos y las operaciones contra accesos no autorizados. Ejemplo: "El sistema debe encriptar todos los datos sensibles con un estándar AES de 256 bits."
3. **Usabilidad:** Garantiza que los usuarios puedan interactuar con el sistema de manera intuitiva y eficiente. Ejemplo: "La plataforma debe ser accesible desde dispositivos móviles y cumplir con las normas de accesibilidad WCAG 2.1."
4. **Escalabilidad:** Asegura que el sistema pueda manejar un aumento en la carga de trabajo. Ejemplo: "El sistema debe soportar hasta 10,000 usuarios concurrentes sin degradar el rendimiento."

Ejemplos Prácticos:

- **En una aplicación bancaria, un requisito de seguridad puede ser** : "El sistema debe implementar autenticación multifactor para los usuarios."
- **En un portal educativo, un requisito de usabilidad puede ser** : "El diseño de la interfaz debe permitir a los estudiantes acceder a los materiales en menos de tres clics."
- **En una tienda en línea, un requisito de rendimiento puede ser** : "La página de inicio debe cargarse en menos de 1 segundo bajo condiciones normales."

2.3. Requisitos técnicos

Los requisitos técnicos especifican las condiciones tecnológicas y las características del sistema necesarias para que el software funcione correctamente. Estos requisitos establecen el marco técnico en el que se desarrollará y operará el sistema, y son fundamentales para garantizar su compatibilidad, escalabilidad y mantenimiento.

Dependencias tecnológicas:

Las dependencias tecnológicas hacen referencia a las plataformas, frameworks, lenguajes de programación y herramientas necesarias para implementar el sistema. Por ejemplo:

Documento solo para uso personal. Prohibido copiar o distribuir

- **Lenguaje de programación**: "El sistema debe desarrollarse en Python 3.10 o superior."
- **Base de datos:** "Se utilizará PostgreSQL como sistema de gestión de bases de datos relacionales."
- **Servidores:** "El sistema debe desplegarse en un entorno AWS con instancias EC2."
- **Integraciones**: "El sistema debe conectarse con una API externa para procesar pagos mediante Stripe."

Especificaciones del sistema:

Estas describen las características técnicas necesarias para el correcto funcionamiento del software, como los requisitos de hardware, sistema operativo y capacidad de red. Ejemplos:

- **Hardware**: "El servidor debe contar con 16 GB de RAM y procesadores de 8 núcleos."
- **Sistema operativo**: "El sistema debe ejecutarse en Linux Ubuntu 22.04."
- **Red:** "La aplicación debe operar en redes con una velocidad mínima de 10 Mbps."
- **Compatibilidad**: "El software debe ser compatible con navegadores modernos como Chrome, Firefox y Safari."

Importancia:

Los requisitos técnicos son esenciales para que los desarrolladores y administradores de sistemas tengan claras las condiciones en las que debe operar el producto. Además, facilitan la planificación del despliegue, reduzcen riesgos relacionados con incompatibilidades y aseguran que el sistema esté alineado con las infraestructuras tecnológicas del cliente. Una definición clara de estos requisitos es clave para un desarrollo eficiente y un producto funcional.

2.4. Requisitos legales y normativos

Los requisitos legales y normativos son esenciales en el desarrollo de software, ya que garantizan que el sistema cumpla con las leyes, regulaciones y estándares específicos aplicables en su contexto de uso. Su incumplimiento puede resultar en sanciones legales, pérdida de confianza de los usuarios y problemas de reputación.

Cumplimiento de regulaciones:
El software debe adaptarse a las normativas locales, nacionales e internacionales que regulen aspectos como la privacidad, la seguridad de los datos y los derechos de los usuarios. Ejemplos comunes incluyen:
Protección de datos personales:
Normativas como el RGPD (Reglamento General de Protección de Datos) en Europa exigen que el software implemente medidas como la anonimización de datos y el consentimiento explícito del usuario para su recopilación y uso.
Accesibilidad:
Leyes como la ADA (Americans with Disabilities Act) o las WCAG (Web Content Accessibility Guidelines) establecen que los sistemas deben ser accesibles para personas con discapacidades.

campus.euniv.eu © Universitas Europaea IMF Juan Ulises PÉREZ VISAIRAS

Normativas específicas por sector:

En sectores como el sanitario, financiero o gubernamental, existen regulaciones estrictas que deben cumplirse:

Sanitario:

La normativa HIPAA en EE.UU. regula cómo se manejan los datos de pacientes en aplicaciones de salud.

Financiero:

Normas como PCI-DSS exigen medidas de seguridad para la gestión de pagos y transacciones financieras.

Importancia del cumplimiento:

Garantizar que el software cumpla con estas normativas no solo evita problemas legales, sino que también aumenta la confianza de los usuarios en el producto. Para lograrlo, es fundamental realizar auditorías de conformidad durante el desarrollo y establecer procesos de revisión que incluyan la validación de los requisitos legales y normativos.

En conclusión, los requisitos legales y normativos son un pilar crucial para asegurar que el software sea seguro, accesible y ético, además de cumplir con las expectativas legales de los stakeholders.

3. Fases de la Ingeniería de Requisitos

La ingeniería de requisitos se organiza en varias fases que garantizan un proceso estructurado y efectivo para definir, analizar y gestionar los requisitos de un proyecto. Estas fases incluyen la recolección de información, el análisis para resolver conflictos y priorizar necesidades, la especificación mediante documentación clara y estandarizada, y la validación para asegurar que los requisitos cumplan con las expectativas de los stakeholders. Cada fase es esencial para minimizar riesgos, evitar errores en etapas avanzadas y garantizar que el producto final esté alineado con los objetivos del cliente y del equipo de desarrollo.

3.1. Recolección:

La recolección de requisitos es la primera fase de la ingeniería de requisitos y tiene como objetivo identificar y capturar las necesidades, expectativas y restricciones de los stakeholders. Esta etapa es crucial, ya que una recolección incompleta o inexacta puede llevar a malentendidos y problemas durante el desarrollo del software.

Importancia de la recolección

Una recolección bien planificada asegura que el equipo técnico tenga una comprensión clara de los requisitos, minimizando cambios inesperados y aumentando la probabilidad de éxito del proyecto. Las técnicas y herramientas utilizadas deben adaptarse al contexto del proyecto y a las características de los stakeholders involucrados.

Documento solo para uso personal. Prohibido copiar o distribuir

Técnicas para recopilar información:

- **Entrevistas:** Reuniones individuales o grupales con stakeholders clave para entender sus necesidades y objetivos. Las entrevistas pueden ser estructuradas (preguntas predefinidas) o abiertas (más flexibles).
- **Encuestas:** Métodos eficaces para recopilar datos de un gran número de personas. Son útiles para identificar patrones generales o validar hipótesis.
- **Talleres:** Sesiones colaborativas donde los stakeholders y el equipo técnico trabajan juntos para definir los requisitos. Los talleres fomentan la comunicación, la resolución de conflictos y la alineación de expectativas.

Herramientas para gestionar la recolección :

- **User Stories (Historias de Usuario):** Una técnica centrada en el usuario para describir las funcionalidades del sistema desde su perspectiva. Por ejemplo: "Como estudiante, quiero acceder a mis calificaciones para conocer mi progreso."
- **Prototipos:** Versiones preliminares del sistema, como maquetas o simulaciones, que permiten a los stakeholders visualizar cómo funcionará el software y proporcionar retroalimentación antes de la implementación.
- **Mapas mentales:** Diagramas visuales que organizan ideas y muestran relaciones entre diferentes conceptos, ayudando a clarificar y priorizar requisitos.

3.2. Análisis:

La fase de análisis en la ingeniería de requisitos tiene como objetivo examinar la información recopilada para identificar inconsistencias, conflictos o redundancias y priorizar los requisitos en función de las necesidades del proyecto y los stakeholders. Este paso es crucial para garantizar que el producto final cumpla con los objetivos definidos.

El análisis de requisitos asegura que el equipo trabaje en las funcionalidades más importantes y elimina problemas antes de avanzar a las fases de diseño e implementación, mejorando la eficiencia y la calidad del proyecto.

Identificación de requisitos inconsistentes o conflictivos:

Durante esta etapa, se busca resolver discrepancias entre las necesidades expresadas por diferentes stakeholders o detectar requisitos contradictorios. Por ejemplo, un usuario podría solicitar simplicidad en la interfaz, mientras que otro exige funcionalidades avanzadas que complican su diseño. Para abordar estos conflictos, es esencial mantener una comunicación abierta y documentar las decisiones tomadas. Además, el análisis ayuda a eliminar requisitos redundantes o que no aportan valor al producto, optimizando recursos y tiempo.

Priorización de requisitos: técnicas y criterios:

No todos los requisitos tienen el mismo nivel de importancia, por lo que es necesario establecer prioridades. Algunas técnicas comunes son:

1. **MoSCoW (Must, Should, Could, Won't):** Clasifica los requisitos en cuatro categorías según su importancia para el éxito del proyecto.
2. **Análisis de valor:** Evalúa el impacto de cada requisito en términos de beneficio para el usuario y el costo de su implementación.

Documento solo para uso personal. Prohibido copiar o distribuir

3. **Priorización por ranking:** Los stakeholders asignan un nivel de prioridad a cada requisito en una escala definida.

Los criterios más comunes para priorizar incluyen:

- **Impacto:** Cuánto contribuye el requisito al objetivo del proyecto.
- **Esfuerzo:** Recursos necesarios para implementarlo.
- **Urgencia:** Plazos asociados al requisito.

3.3. Especificación

La especificación de requisitos es una fase crucial en la ingeniería de requisitos, ya que consiste en documentar de manera clara, precisa y estructurada las necesidades y características del sistema. Este documento sirve como referencia para el equipo técnico, los stakeholders y cualquier otra parte interesada durante el desarrollo del proyecto.

El documento de especificación garantiza que todos los involucrados compartan una visión común del sistema, minimizando malentendidos y errores en etapas posteriores del proyecto.

Documentación de requisitos: plantillas y estándares:

La especificación de requisitos debe seguir plantillas o estándares establecidos que aseguren la consistencia y claridad de la información. Algunos de los estándares más comunes son:

- **IEEE 830:** Proporciona pautas para la creación de un Documento de Especificación de Requisitos de Software (SRS), organizando los requisitos en secciones como alcance, definiciones, requisitos funcionales y no funcionales, y restricciones.
- **Plantillas personalizadas:** Empresas y equipos pueden desarrollar sus propias plantillas, adaptadas a las necesidades específicas del proyecto o cliente.

Una buena especificación debe incluir:

- **Requisitos funcionales y no funcionales:** Descripciones detalladas de las funcionalidades esperadas y las características de calidad.
- **Casos de uso:** Escenarios de interacción del usuario con el sistema.
- **Diagramas y gráficos:** Representaciones visuales como diagramas UML para aclarar relaciones y flujos.

Ejemplo de un documento de especificación:

1. **Introducción:** Objetivos, alcance del sistema y definiciones clave.
2. **Descripción general:** Resumen del contexto del sistema, actores involucrados y dependencias tecnológicas.
3. **Requisitos funcionales:** "El sistema debe permitir a los usuarios iniciar sesión con autenticación de dos factores."
4. **Requisitos no funcionales:** "El sistema debe procesar 500 solicitudes por segundo con un tiempo de respuesta inferior a 2 segundos."
5. **Restricciones:** Normativas aplicables, como el cumplimiento del RGPD.

3.4. Validación:

Documento solo para uso personal. Prohibido copiar o distribuir

La validación de requisitos es una fase crucial en la ingeniería de requisitos que asegura que las necesidades y expectativas de los stakeholders estén correctamente representadas y documentadas. El objetivo principal es verificar que los requisitos sean claros, consistentes, completos y alcanzables antes de avanzar a las etapas de diseño e implementación.

La validación no solo mejora la calidad de los requisitos, sino que también reduce riesgos y costos asociados a cambios en etapas avanzadas. Implementar métodos efectivos y fomentar la retroalimentación garantiza que los requisitos sean una base sólida para el éxito del proyecto.

Métodos para verificar y validar requisitos

1. **Revisiones y reuniones:** Reuniones con los stakeholders para revisar los requisitos, identificar errores, aclarar dudas y resolver inconsistencias.
2. **Prototipos:** Creación de modelos preliminares que permiten a los usuarios visualizar cómo se implementarán los requisitos, ayudando a identificar mejoras o problemas.
3. **Casos de prueba:** Desarrollo de pruebas específicas para verificar que cada requisito pueda ser implementado y cumplido.
4. **Análisis de trazabilidad:** Comprobación de que todos los requisitos están vinculados a objetivos del proyecto, evitando requisitos no alineados.

Importancia de la retroalimentación con stakeholders

La participación activa de los stakeholders durante la validación es fundamental para garantizar que los requisitos capturados sean correctos y útiles. La retroalimentación permite:

- **Corregir errores tempranos:** Identificar requisitos malinterpretados o mal documentados.
- **Alinear expectativas:** Asegurar que los stakeholders y el equipo técnico compartan la misma visión del sistema.
- **Facilitar la aceptación:** Un sistema desarrollado con la colaboración de los stakeholders tiene mayores probabilidades de cumplir con sus expectativas y obtener su aprobación final.

4. Herramientas y Técnicas en la Ingeniería de Requisitos

4.1. Herramientas digitales

En la ingeniería de requisitos, las herramientas digitales desempeñan un papel fundamental para gestionar de manera eficiente los requisitos y garantizar la comunicación entre los stakeholders y el equipo técnico. Estas herramientas ayudan a organizar, documentar, priorizar y realizar un seguimiento de los requisitos a lo largo del ciclo de vida del proyecto.

Jira:
Es una de las herramientas más populares para la gestión de proyectos ágiles. Con Jira, los equipos pueden organizar los requisitos en forma de tickets o historias de usuario, asignarlos a los miembros del equipo y realizar un seguimiento de su progreso. Además, permite priorizar tareas y visualizar el avance a través de tableros Kanban o Scrum.

campus.eunivi.eu © Universitas Europaea . Juan Ulises PÉREZ VISAIRAS

Trello:

Es una herramienta sencilla y visual que utiliza tableros y tarjetas para gestionar tareas. Aunque no está diseñada específicamente para la gestión de requisitos, es ideal para proyectos pequeños o equipos que buscan una solución simple para organizar y priorizar tareas. Otra de características muy similares es MS Planner

Herramientas específicas de gestión de requisitos:

- **DOORS:** Utilizada en proyectos complejos, especialmente en sectores como la ingeniería y la automoción, permite documentar, rastrear y gestionar requisitos detallados.
- **ReqView:** Una solución centrada en la trazabilidad y gestión de requisitos, que facilita mantener una relación clara entre los requisitos y los entregables del proyecto.

Estas herramientas no solo mejoran la organización y la colaboración, sino que también reducen riesgos asociados a errores o malentendidos, asegurando que los requisitos estén alineados con los objetivos del proyecto.

4.2. Técnicas visuales

Las técnicas visuales son herramientas esenciales en la ingeniería de requisitos, ya que permiten representar información compleja de manera clara y comprensible. Facilitan la comunicación entre los stakeholders y el equipo técnico, mejorando la comprensión y el análisis de los requisitos.

Diagramas de casos de uso

Estos diagramas muestran cómo los usuarios (actores) interactúan con el sistema para lograr objetivos específicos. Incluyen actores, casos de uso (acciones o procesos) y sus relaciones. Por ejemplo, un caso de uso para una tienda en línea podría incluir "Buscar productos" y "Realizar pedido". Los diagramas de casos de uso son ideales para identificar y organizar requisitos funcionales.

Diagramas UML

El Lenguaje Unificado de Modelado (UML) proporciona múltiples diagramas, como los de clases, secuencia y actividades, que representan diferentes aspectos del sistema. Por ejemplo:

- **Diagrama de clases:** Define las entidades del sistema, sus atributos y relaciones.
- **Diagrama de secuencia:** Muestra la interacción entre objetos a lo largo del tiempo. Estos diagramas ayudan a detallar el diseño técnico y a garantizar que los requisitos se traduzcan correctamente en soluciones implementables.

Mapas mentales

Documento solo para uso personal. Prohibido copiar o distribuir

Son diagramas jerárquicos que organizan ideas y conceptos relacionados con los requisitos. Por ejemplo, un mapa mental puede mostrar los módulos principales de un sistema, sus funcionalidades y dependencias. Son útiles para la lluvia de ideas y la priorización de requisitos.

Estas técnicas visuales no solo facilitan el análisis y la especificación de los requisitos, sino que también promueven la colaboración y alineación entre todos los involucrados en el proyecto.

4.3. Prototipado

El prototipado es una técnica clave en la ingeniería de requisitos que consiste en crear representaciones preliminares del sistema, como maquetas o prototipos funcionales. Su objetivo es clarificar los requisitos al ofrecer una visión tangible de cómo podría ser el producto final. Esto facilita la comunicación entre los stakeholders y el equipo técnico, reduce malentendidos y mejora la precisión en la definición de los requisitos.

Creación de maquetas

Las maquetas son representaciones estáticas del sistema que muestran la estructura, el diseño y la disposición de los elementos de la interfaz. Herramientas como Figma, Adobe XD o incluso presentaciones simples se utilizan para crear estas visualizaciones. Las maquetas ayudan a los stakeholders a comprender cómo será la experiencia de usuario y permiten realizar ajustes tempranos en el diseño.

Prototipos funcionales

Los prototipos funcionales son versiones interactivas y simplificadas del sistema que permiten a los stakeholders probar funcionalidades básicas. Esto incluye navegación entre pantallas, simulación de procesos y validación de interacciones. Herramientas como Axure o herramientas de desarrollo rápido permiten crear prototipos que simulan el comportamiento del sistema.

Ventajas del prototipado:

- Detecta y corrige errores en las primeras etapas.
- Facilita la retroalimentación de los stakeholders.
- Reduce riesgos y costos asociados a cambios tardíos.

El prototipado es especialmente útil en proyectos donde los requisitos no están completamente definidos, permitiendo iterar hasta llegar a una solución que satisfaga tanto las expectativas del cliente como las capacidades técnicas del equipo.

5. Desafíos y Soluciones en la Ingeniería de Requisitos

5.1. Gestión de cambios en los requisitos

La gestión de cambios en los requisitos es un proceso crítico en la ingeniería de requisitos, ya que asegura que las modificaciones realizadas durante el desarrollo del proyecto sean evaluadas, aprobadas y documentadas de manera adecuada. Los cambios en los requisitos pueden surgir por varias razones, como la evolución de las necesidades del cliente, cambios en el mercado, o descubrimientos técnicos durante el desarrollo.

campus.euniv.eu © Universitas Europaea IMF Juan Ulises PÉREZ VISAIRAS

Pasos para la gestión de cambios:

1. **Identificación del cambio:** Documentar el cambio propuesto y sus razones. Esto puede incluir nuevas funcionalidades, ajustes en las existentes o eliminación de requisitos obsoletos.
2. **Evaluación del impacto:** Analizar cómo afectará el cambio al alcance, costo, tiempo y recursos del proyecto. Herramientas como diagramas de trazabilidad pueden ayudar a evaluar las dependencias afectadas.
3. **Aprobación:** Los cambios deben ser revisados y aprobados por un comité o los stakeholders relevantes para garantizar que sean necesarios y beneficiosos.
4. **Implementación:** Una vez aprobado, el cambio se incorpora al proyecto y se actualizan los documentos de requisitos.

Importancia de la gestión de cambios

Sin una gestión adecuada, los cambios pueden generar retrasos, sobrecostos y confusión en el equipo. Implementar procesos claros y utilizar herramientas de seguimiento como Jira o Trello asegura que los cambios sean manejados de manera eficiente, minimizando riesgos y manteniendo el proyecto alineado con los objetivos del cliente.

5.2. Comunicación con stakeholders diversos

La comunicación efectiva con los stakeholders es crucial en la ingeniería de requisitos, ya que garantiza que todas las partes involucradas comprendan y acuerden las necesidades del sistema. Los stakeholders tienen roles y perspectivas diversas, lo que hace necesario adaptar el enfoque de comunicación según sus intereses y responsabilidades. Entender y comunicar eficazmente con estos roles asegura que todos los stakeholders estén alineados, se eviten malentendidos y se logre un producto final que cumpla con las expectativas de todas las partes involucradas.

Roles principales de los stakeholders:

Patrocinador
Es el encargado de financiar el proyecto y, generalmente, un alto directivo de la organización. Su enfoque está en los objetivos estratégicos, el retorno de inversión y el cumplimiento de plazos y presupuestos. La comunicación con el patrocinador debe centrarse en resultados generales, avances significativos y cómo el proyecto contribuirá al éxito de la organización.
Director funcional
Este rol representa las áreas de negocio que interactúan con el sistema. El director funcional prioriza que el software se alinee con los procesos de negocio y cumpla con las expectativas operativas. La comunicación debe incluir detalles sobre cómo los requisitos reflejan las necesidades del negocio y optimizan su funcionamiento.
Usuario directo
Son los usuarios finales que utilizarán el sistema en su día a día. Para ellos, es esencial que el sistema sea funcional, accesible y fácil de usar. La comunicación debe incluir demostraciones prácticas, prototipos y consultas frecuentes para asegurar que sus necesidades sean comprendidas y satisfechas.

Documento solo para uso personal. Prohibido copiar o distribuir

5.3. Equilibrio entre requisitos funcionales y no funcionales

Lograr un equilibrio entre los requisitos funcionales y no funcionales es fundamental para el éxito de un proyecto de software. Mientras que los requisitos funcionales definen las funcionalidades específicas que debe realizar el sistema (como autenticación de usuarios o generación de informes), los no funcionales establecen cómo debe comportarse el sistema (rendimiento, seguridad, usabilidad, etc.). Ambos tipos son complementarios y deben trabajarse en conjunto.

Importancia del equilibrio

Enfocarse exclusivamente en los requisitos funcionales puede llevar a un sistema que cumple con las funcionalidades esperadas pero que no es eficiente, seguro o fácil de usar. Por otro lado, priorizar únicamente los requisitos no funcionales puede resultar en un sistema bien optimizado pero que carece de las características fundamentales necesarias para los usuarios.

Estrategias para lograr el equilibrio

1. **Priorización conjunta:** Usar técnicas como MoSCoW para identificar la importancia relativa de los requisitos funcionales y no funcionales.
2. **Evaluación iterativa:** Revisar continuamente cómo los requisitos funcionales afectan a los no funcionales y viceversa.
3. **Validación temprana:** Implementar pruebas que aseguren que los requisitos no funcionales, como el tiempo de respuesta, no se vean afectados por funcionalidades adicionales.

Este equilibrio permite entregar un sistema que no solo satisface las necesidades del usuario, sino que también garantiza una experiencia óptima, contribuyendo al éxito y sostenibilidad del proyecto.

Bibliografía y lecturas recomendadas:

- **Pressman, R. S., & Maxim, B. R. (2015).** *Ingeniería del Software: Un Enfoque Práctico.* McGraw-Hill.
- **Sommerville, I. (2011).** *Ingeniería de Software.* Pearson.
- **Grau, G., & Franch, X. (2008).** *Requisitos: El arte de analizar y documentar.* UOC.
- **Mateos García, S. (2013).** *Gestión de requisitos en proyectos de software.* Ra-Ma.
- **Manifiesto Ágil:** https://agilemanifesto.org/iso/es/
- **IEEE Computer Society**: https://www.computer.org/
- **Requisitos.com**: https://www.requisitos.com/

Documento solo para uso personal. Prohibido copiar o distribuir

Actividades prácticas

Ejercicio 5. Gestión de Requisitos para el Desarrollo de un Sistema de Reservas de Consultas Médicas

Una clínica desea desarrollar un sistema de reservas en línea que permita a los pacientes:

1. Registrar una cuenta y autenticarse.
2. Seleccionar un médico y reservar una cita según la disponibilidad.
3. Recibir notificaciones de confirmación y recordatorio por correo electrónico.
4. Consultar su historial de citas.

El sistema debe cumplir con las siguientes condiciones:

- Proteger los datos de los pacientes según el RGPD.
- Procesar hasta 500 solicitudes por segundo.
- Ser accesible desde dispositivos móviles y cumplir con las normas WCAG 2.1.

1. Identifica y clasifica los requisitos funcionales, no funcionales, técnicos y legales del sistema. Proporciona un caso de uso para una funcionalidad clave del sistema.
Propuesta de validación: explica cómo validarías los requisitos para asegurar que cumplen con las expectativas del cliente.

Procesando respuesta, no cierres el navegador, este proceso podría tardar unos segundos

Ejercicio 6. Sistema de Gestión de Pedidos para una Tienda Online

Una tienda online desea desarrollar un sistema que permita a los clientes:

1. Registrarse y acceder a su cuenta.
2. Navegar por categorías de productos y añadirlos al carrito.
3. Realizar pagos seguros mediante una pasarela de pago integrada.
4. Recibir confirmación del pedido por correo electrónico.

El sistema debe cumplir con los siguientes requisitos:

- Procesar hasta 1000 pedidos simultáneos.
- Cargar la página principal en menos de 2 segundos.
- Cumplir con el RGPD para garantizar la seguridad de los datos de los clientes.

Documento solo para uso personal. Prohibido copiar o distribuir

1. Clasifica los requisitos funcionales, no funcionales, técnicos y legales del sistema.
   Proporciona un caso de uso para la funcionalidad de realizar un pedido.
   Explica cómo validarías los requisitos para garantizar que se alineen con las necesidades del cliente.

Procesando respuesta, no cierres el navegador, este proceso podría tardar unos segundos

Documento solo para uso personal. Prohibido copiar o distribuir