Tema 4: Análisis de Requisitos

BLOQUE II: ESPECIFICACIÓN DE REQUISITOS Y ANÁLISIS DE LOS SISTEMAS SOFTWARE

Ingeniería del Software
Grado en Ingeniería Informática
Curso 2024/2025

David Cáceres Gómez

UNIVERSIDAD
D
CÓRDOBA

Índice

1. Introducción

Problemática

"La correcta obtención de los requisitos es uno de los aspectos más críticos de un proyecto software, independientemente del tipo de proyecto que se trate, dado que una mala captura de los mismos es la causa de la mayor parte de los problemas que surgen a lo largo del ciclo de vida" [Johnson, 1995]

Problemas en la obtención de requisitos

"La parte más difícil de construir de un sistema software es decidir qué construir. [...] Ninguna otra parte del trabajo afecta más negativamente al sistema final si se realiza de manera incorrecta. Ninguna otra parte es más difícil de rectificar después" [Brooks, 1995]

Crisis del Software

"El coste de un cambio en los requisitos, una vez entregado el producto, es entre 60 y 100 veces superior al coste que hubiera representado el mismo cambio durante las fases iniciales de desarrollo" [Pressman, 2002]

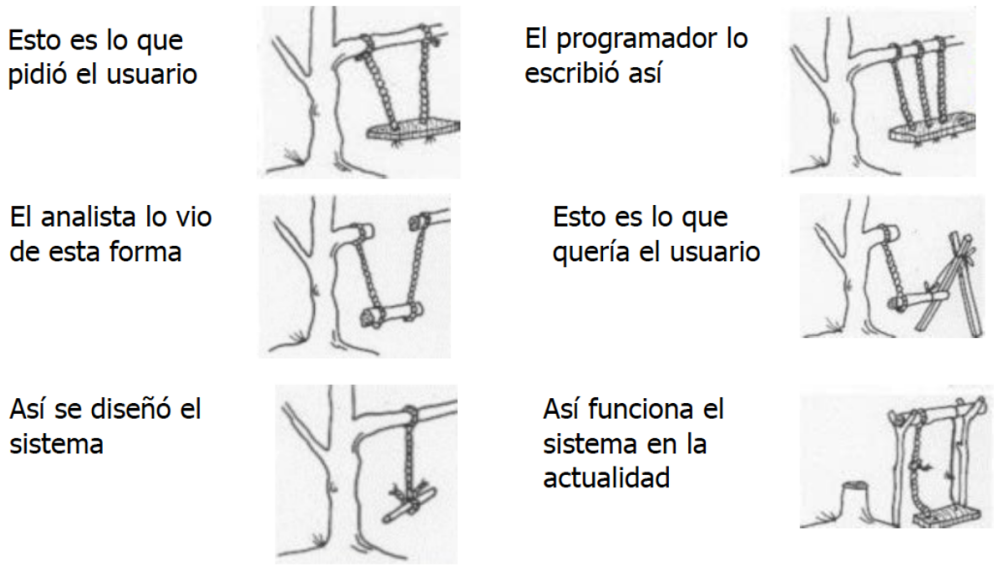

Figura 1: Problemática [1]

Primera fase del ciclo de vida del software en la que se produce una especificación a partir de ideas informales. Deben obtenerse y documentarse:

- Los requisitos de información
- Los requisitos funcionales
- Los requisitos no funcionales
- Los criterios para medir el grado de su consecución

El proceso de desarrollo de dicha especificación de requisitos es lo que se conoce como Ingeniería de Requisitos.

2. Definición de Requisitos

Definición

Un requisito es una característica **observable** del sistema que **satisface una necesidad** o expresa una restricción que afecta al software que estamos desarrollando.

¿Qué son los Requisitos?

Describen los servicios que el sistema debe ofrecer, así como sus restricciones operativas. Un requisito puede variar desde una declaración abstracta de alto nivel hasta una especificación formal y detallada de una función del sistema. Es importante distinguir entre diferentes niveles de descripción:

- Requisitos del Sistema
- Requisitos de Usuario

La comunicación juega un papel fundamental en la ingeniería de requisitos, lo que añade complejidad a esta disciplina, ya que interviene el factor humano. Este factor introduce aspectos sociales y culturales, no solo técnicos.

¿Qué describe un requisito?

- Una utilidad para el usuario "El tratamiento de textos ha de incluir la comprobación y corrección gramatical"

- Una propiedad general del sistema "El sistema ha de garantizar que la información personal solamente será accesible mediante autorización explícita"
- Una restricción general del sistema "El sensor ha de muestrearse 10 veces por segundo"
- Cómo llevar a cabo cierto cálculo "Calificación final = nota examen + 2*nota trabajo + 2/3 nota ejercicios"
- Una restricción sobre el desarrollo del sistema "El sistema ha de implementarse en C++"

Requisitos vs Restricciones

Dada una lista de deseos y necesidades, ¿cómo se distinguen los requisitos de las restricciones? Las restricciones limitan las posibles soluciones a un problema.

Por ejemplo, el programa debe imprimir transparencias de color en blanco y negro, ajustando automáticamente la correspondencia de color, y utilizando una impresora *PostScript*.

- **Rq1:** Imprimir transparencias de color en una impresora de blanco y negro.
- **Rq2:** Establecer la correspondencia al blanco y negro de forma automática.
- **Restricción:** Utilizar el formato *PostScript*.

Es importante señalar que lo que es tangible o visible para el usuario generalmente se considera un requisito. En este caso, los dos primeros son visibles para el usuario, mientras que el tercero no lo es.

Relevancia de los requisitos

Una definición inadecuada de los requisitos puede generar problemas. Requisitos ambiguos pueden ser interpretados de manera distinta por usuarios y desarrolladores. Por ejemplo, el término "visores apropiados" podría interpretarse de las siguientes formas:

- **Intención del Usuario:** Visores especializados para cada tipo de documento.
- **Interpretación del Desarrollador:** Un visor de texto que muestre el contenido del documento.

Idealmente, los requisitos deberían ser tanto completos como consistentes:

- **Completos:** Recogen la descripción de todos los servicios esperados por el usuario.
- **Consistentes:** No deben existir contradicciones entre las especificaciones.

Sin embargo, en la práctica, es casi imposible crear un documento de requisitos que sea completamente completo y consistente.

Consejos para la definición de requisitos

Para facilitar la claridad y la comprensión en la definición de requisitos, es importante seguir algunas prácticas recomendadas. En primer lugar, se sugiere establecer un formato más o menos estándar y adherirse a él para todos los requisitos.

Asimismo, es fundamental utilizar el lenguaje de manera consistente:

- **Uso del presente** para los requisitos obligatorios, lo que indica que estas características son imprescindibles para el sistema.
- **Uso del modo condicional** para los requisitos deseables, que indica que estas características son útiles, pero no imprescindibles.

Además, se recomienda subrayar las partes relevantes del texto para destacar aspectos clave y facilitar la rápida identificación de información crucial.

Finalmente, es esencial evitar el uso de jerga técnica o términos que puedan no ser familiares para todos los interesados, ya que esto puede llevar a malentendidos y a la falta de claridad en los requisitos.

3. Tipos de Requisitos

Los requisitos en un proyecto de desarrollo se dividen principalmente en dos grandes categorías: requisitos del sistema y requisitos del usuario. A continuación, se explicarán en mayor detalle ambos tipos de requisitos.

3.1. Requisitos del Sistema

Requisitos del Sistema (descripción detallada del sistema): Estos requisitos describen en detalle las funciones, servicios y restricciones operativas que debe cum-

plir el sistema. Son específicos y deben definir con exactitud lo que se implementará, ya que pueden formar parte del contrato entre el comprador y el desarrollador. Los requisitos del sistema se dividen en varios tipos:

- Requisitos Funcionales
- Requisitos No Funcionales:
  - Requisitos del Producto
  - Requisitos Organizacionales
  - Requisitos Externos

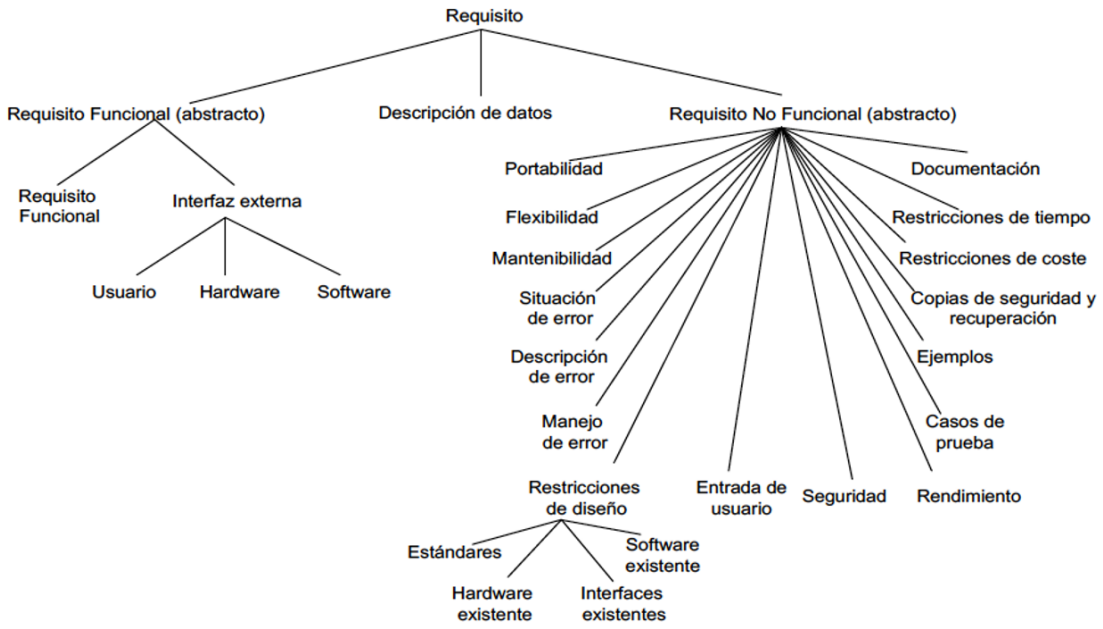

Figura 2: Jerarquía de especialización de RSM [Pohl, 1997]

3.1.1. Requisitos Funcionales

Los **Requisitos Funcionales** se refieren a las funciones y necesidades que el sistema debe satisfacer, es decir, lo que debe hacer. Los requisitos funcionales describen los cálculos que realiza el sistema, los datos que maneja y cómo los procesa.

En términos generales, estos requisitos especifican cómo debe comportarse el sistema en respuesta a los estímulos externos.

3.1.2. Requisitos No Funcionales

Los **Requisitos No Funcionales** no describen funciones específicas del sistema, sino que **imponen restricciones** sobre los servicios o funciones que este ofrece, como la fiabilidad, tiempos de respuesta o capacidad de almacenamiento.

Otra diferencia respecto a los requisitos funcionales es que los requisitos no funcionales **no incluyen comportamiento**. Su finalidad es especificar criterios que evalúen la calidad general del sistema, como seguridad, rendimiento, coste, etc.

Suelen aplicarse al sistema en su conjunto y derivan de las necesidades del usuario, como limitaciones presupuestarias, políticas organizacionales o requisitos de interoperabilidad.

Estos requisitos definen propiedades y restricciones del sistema, incluyendo aspectos como tiempos de respuesta y necesidades de almacenamiento. También pueden referirse al proceso de desarrollo, como el uso de herramientas CASE, lenguajes de programación o metodologías específicas.

En general, son más críticos que los requisitos funcionales, ya que, si no se cumplen, el sistema puede no ser útil.

Los tipos de requisitos no funcionales incluyen:

- **Requisitos del Producto:** Especifican el comportamiento del producto.

  Ejemplos: rapidez de la ejecución, capacidad de memoria, fiabilidad, etc.

- **Requisitos Organizacionales:** Derivan de políticas y procedimientos existentes en la organización del cliente y del desarrollador.

  Ejemplos: Estándares de procesos, métodos de diseño, lenguajes de programación, métodos de entrega, etc.

- **Requisitos Externos**: Se derivan de factores externos al sistema y a sus procesos de desarrollo.

  Ejemplos: Requisitos de interoperabilidad, legislativos, éticos, etc.

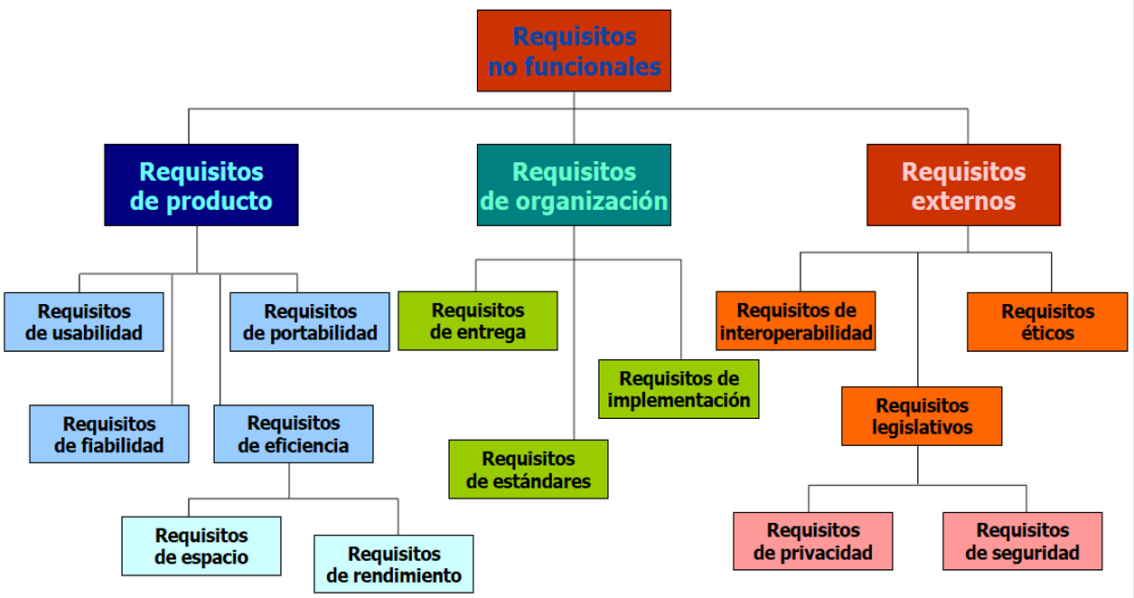

Figura 3: Requisitos no funcionales [2]

Metas y Requisitos

Cuando se trata de requisitos no funcionales, puede ser difícil definirlos con precisión. Los requisitos imprecisos resultan complicados de verificar.

- **Meta:** Un propósito general del usuario, como la facilidad de uso.
- **Requisito no funcional verificable:** Una declaración que incluye una medida objetiva.

Las metas son útiles para los desarrolladores, ya que reflejan los deseos generales del usuario.

- **Ejemplo de Meta:** El sistema debe ser fácil de usar para usuarios cualificados y debe estar diseñado para minimizar posibles errores del usuario.
- **Ejemplo de Requisito No Funcional Verificable:** Los usuarios cualificados deben ser capaces de utilizar todas las funciones del sistema tras 2 horas de entrenamiento. El promedio de errores por usuario no debe superar los 2 errores diarios después de dicho entrenamiento.

3.2. Requisitos del Usuario

Requisitos del Usuario (Descripción de alto nivel): Son descripciones, ya sea en lenguaje natural o mediante diagramas, que detallan lo que se espera que el sistema proporcione, así como las restricciones en las que debe operar.

Estos requisitos deben incluir tanto requisitos funcionales como no funcionales y deben redactarse de manera que sean comprensibles para los usuarios del sistema que no tienen conocimientos técnicos. Se pueden presentar utilizando lenguaje natural, tablas y diagramas.

Problemas Comunes:

- **Falta de claridad:** Es difícil ser preciso sin que el documento se vuelva complicado de leer.
- **Mezcla de tipos de requisitos:** A menudo, se combinan requisitos funcionales y no funcionales, lo que puede generar confusión.
- **Fusión de requisitos:** Es habitual agrupar varios requisitos en la definición de uno solo, lo que puede dificultar la identificación de necesidades específicas.

Ejemplo: "El sistema LIBSYS debe ofrecer un subsistema de contabilidad que mantenga un registro de todos los pagos realizados por los usuarios. Además, los administradores deben tener la capacidad de configurar este subsistema para ofrecer descuentos a los usuarios frecuentes."

3.3. Requisitos de Interfaz

Dependiendo de la bibliografía consultada, podemos encontrar otros tipos de requisitos, como los requisitos de interfaz.

Los requisitos de interfaz describen cómo la aplicación interactúa con su entorno, tanto en términos de comunicación con los usuarios como con otras aplicaciones o sistemas. Estos requisitos son cruciales para garantizar que la aplicación sea intuitiva, eficiente y capaz de integrarse sin problemas con otros componentes del sistema.

Ejemplo (comunicación con usuarios): "El coste del envío se mostrará en la esquina inferior derecha."

Ejemplo (comunicación con otras aplicaciones): "Los pedidos se transmitirán en formato XML utilizando la plantilla DTD detallada en el anexo IV."

4. Gestión de Requisitos

Tras la identificación de los requisitos, estos deben seguir un proceso estructurado que incluya las siguientes etapas:

- **Incorporación en un Catálogo:** Todos los requisitos deben ser documentados en un catálogo accesible, donde se puedan consultar fácilmente. Este catálogo sirve como una base de datos centralizada que permite a todos los miembros del equipo de desarrollo y partes interesadas tener acceso a la información relevante.
- **Análisis Exhaustivo:** Los requisitos deben ser analizados minuciosamente para identificar posibles inconsistencias, ambigüedades, duplicidades o falta de información. Este análisis es esencial para asegurar que los requisitos sean claros y que no haya interpretaciones erróneas que puedan afectar el desarrollo del sistema.
- **Validación con Clientes/Usuarios:** Es crucial que los requisitos sean validados por los clientes o usuarios finales para asegurarse de que reflejan sus necesidades y expectativas. Esta validación garantiza que el producto final cumpla con lo que se espera y que se minimicen los cambios posteriores en el desarrollo.

Además de las etapas anteriores, es importante realizar actividades que aseguren que los requisitos se están cumpliendo a lo largo del ciclo de vida del proyecto. Esto incluye prácticas de trazabilidad y coherencia, que permiten rastrear cada requisito desde su origen hasta su implementación, asegurando que todos los aspectos del proyecto estén alineados con los requisitos iniciales.

Características de un Requisito

Cada requisito debe poseer ciertas características que faciliten su gestión y seguimiento. Estas incluyen:

- **Identificador:** Un código único para cada requisito.
- **Autor:** La persona responsable de redactar o proponer el requisito.
- **Tipo de requisito:** Clasificación del requisito (por ejemplo, funcional, no funcional, de usuario, etc.).
- **Descripción:** Un texto claro y conciso que explique el requisito.

- **Prioridad:** Una clasificación de la importancia del requisito (alta, media o baja).
- **Estado:** La fase en la que se encuentra el requisito (propuesto, aprobado, incorporado, etc.).
- **Fecha de Creación/Revisión:** Fechas que indican cuándo fue creado o revisado el requisito, lo que ayuda a mantener el control de versiones.

Organización de un Requisito

Para facilitar su gestión y consulta, los requisitos deben organizarse de manera lógica y estructurada. Algunas formas de organización incluyen:

- **Por Subsistemas:** Agrupar requisitos según los diferentes subsistemas que componen el sistema total.
- **Por Tipo:** Clasificar los requisitos en funcionales (RFX) y no funcionales (RNFX).
- **Jerárquicamente:** Establecer una estructura jerárquica que muestre la relación entre requisitos, como RF1, RF1.1, RF1.1.1, donde cada nivel representa una mayor especificidad.

5. Actividades a Realizar en la Especificación de Requisitos

Las actividades fundamentales involucradas en la especificación de requisitos son las siguientes:

1. **Extracción de Requisitos:** Proceso por el cual los clientes descubren, revelan y comprenden los requisitos que desean.
2. **Análisis de Requisitos:** Proceso de razonamiento sobre los requisitos obtenidos en la etapa anterior, detectando y resolviendo posibles inconsistencias o conflictos, coordinando los requisitos relacionados.
3. **Especificación de Requisitos:** Proceso de redacción o registro de los requisitos. Para este proceso puede recurrirse al lenguaje natural, formal, modelos, gráficos etc.

4. **Validación de Requisitos:** Proceso de confirmación por parte de los usuarios/clientes de los requisitos especificados en el que se comprueba su validez, consistencia, completitud, etc.

La ejecución de estas actividades a menudo se apoya en diversas técnicas. Por ejemplo, para la extracción de requisitos, se utilizan métodos de recolección de información como observación, entrevistas y grupos de trabajo. En las etapas de análisis y especificación, se aplican técnicas gráficas como diagramas de flujo de datos y casos de uso para facilitar la comprensión de los requisitos.

Para la validación, se suelen utilizar listas de verificación junto con técnicas de revisión y evaluación.

Las siguientes técnicas son ampliamente empleadas para la captura y análisis de requisitos:

- **Entrevistas:** Conversaciones estructuradas con usuarios y partes interesadas para obtener información detallada.
- **Desarrollo conjunto de aplicaciones (JAD):** Un enfoque colaborativo que involucra a usuarios, desarrolladores y otros interesados en sesiones de trabajo conjunto.
- **Prototipado:** Creación de modelos iniciales del sistema para facilitar la comprensión de los requisitos y obtener retroalimentación temprana.
- **Observación:** Análisis del entorno en el que se utilizará el sistema para identificar requisitos en función de cómo los usuarios interactúan con sus herramientas actuales.
- **Estudio de documentación:** Revisión de documentos existentes relacionados con el sistema para identificar requisitos y necesidades previas.
- **Cuestionarios:** Herramientas de recopilación de datos que permiten obtener información de un amplio número de usuarios de manera estructurada.
- **Tormenta de ideas (*Brainstorming*):** Sesiones creativas en las que los participantes generan ideas y requisitos sin restricciones.
- **ETHIC (Effective Technical and Human Implementation of Computer-based System):** Un enfoque que se centra en la implementación efectiva tanto de aspectos técnicos como humanos en sistemas computacionales.

6. Especificación de Requisitos del Software (ERS)

El objetivo de la Especificación de Requisitos es generar un documento conocido como Especificación de Requisitos del Software (ERS).

Este documento tiene la finalidad de definir de manera completa, precisa y verificable los requisitos que el sistema debe cumplir, incluyendo tanto los requisitos funcionales como los no funcionales, así como las restricciones aplicables al diseño, tanto de software como de hardware [3]. El contenido de este documento incluye:

1. Introducción
   - 1.1 Objetivo
   - 1.2 Ámbito del Sistema
   - 1.3 Definiciones, Acrónimos y Abreviaturas
   - 1.4 Referencias
   - 1.5 Visión General del Documento
2. Descripción General
   - 2.1 Perspectiva del Producto
   - 2.2 Funciones del Producto
   - 2.3 Características de los Usuarios
   - 2.4 Restricciones
   - 2.5 Suposiciones y Dependencias
3. Requisitos Específicos
   - 3.1 Requisitos Funcionales
     - 3.1.1 Requisito funcional 1
       - 3.1.1.1 Introducción
       - 3.1.1.2 Entradas
       - 3.1.1.3 Salidas
     - 3.1.2 Requisito funcional 2
     - .....
   - 3.2 Requisitos Interfaz Externa
     - 3.2.1 Interfaces de usuario
     - 3.2.2 Interfaces hardware
     - 3.2.3 Interfaces software

Características deseables en una **BUENA** especificación de requisitos:

- **No ambigua:** Los requisitos deben estar redactados de manera clara, evitando cualquier interpretación que pueda generar confusión.
- **Completa:** Deben incluir todos los requisitos necesarios para el funcionamiento del sistema, sin omitir ninguna funcionalidad relevante.
- **Fácil de verificar:** Es esencial que los requisitos puedan ser comprobados de forma sencilla para asegurar que se cumplen en el desarrollo del sistema.
- **Consistente (coherente):** La especificación debe ser interna y externamente coherente, sin contradicciones entre los requisitos.
- **Clasificada por importancia o estabilidad:** Los requisitos deben organizarse según su prioridad o la estabilidad de sus necesidades, facilitando la gestión y el enfoque en los aspectos críticos.
- **Fácil de modificar:** La especificación debe permitir cambios sin complicaciones, adaptándose a nuevas necesidades o correcciones de manera eficiente.
- **Fácil identificación del origen y las consecuencias de cada requisito:** Debe ser sencillo rastrear de dónde proviene cada requisito y cuáles son sus implicaciones en el sistema.

- **De fácil utilización durante la fase de explotación y mantenimiento:** La especificación debe ser práctica y accesible para su uso en las etapas posteriores del ciclo de vida del sistema, como la explotación y el mantenimiento.

Estructura de una Especificación de Requisitos [4]:

- Portada
- Lista de cambios
- Índice
- Lista de figuras
- Lista de tablas

1. Introducción
2. Participantes en el proyecto
3. Descripción del sistema actual
4. Objetivos del sistema
5. Catálogo de requisitos del sistema
   - 5.1 Requisitos de información
   - 5.2 Requisitos funcionales
     - 5.2.1 Diagramas de casos de uso
     - 5.2.2 Definición de actores
     - 5.2.3 Casos de uso del sistema
   - 5.3 Requisitos no funcionales
6. Matriz de rastreabilidad objetivos/requisitos
7. Glosario de términos
8. Conflictos pendientes de resolución [opcional, pueden ir en un documento aparte]

- Apéndices [opcionales]

Usuarios de una Especificación de Requisitos del Software:

- **Clientes:** Especifican los requisitos y los leen para comprobar que estos satisfacen sus necesidades. También especifican los cambios en los requisitos.
- **Gestores:** Utilizan el documento de requisitos para planificar una oferta por el sistema y para planificar el proceso de desarrollo del sistema
- **Desarrolladores:** Usan los requisitos para entender qué es lo que hay que desarrollar.
- **Encargados de pruebas:** Se basan en los requisitos para desarrollar test para la validación del sistema.
- **Encargados de mantenimiento:** Utilizan los requisitos como ayuda para comprender el sistema y las relaciones entre sus partes.

7. Especificación de Requisitos Ágiles

7.1. Historias de Usuario

Las historias de usuario son una herramienta clave en los métodos ágiles para describir de forma general e informal las expectativas del software desde la perspectiva del usuario final. No deben confundirse con los requisitos, ya que las historias de usuario representan un objetivo o resultado deseado, en lugar de una función específica del sistema.

El propósito principal de las historias de usuario es definir cómo la función que se implementará generará valor para el usuario, proporcionando además un contexto claro al equipo de desarrollo. Estas historias están estructuradas de manera narrativa, con el usuario en el centro de la descripción.

Cada historia debe ser breve y fácil de recordar, lo que facilita su escritura en una tarjeta o un *post-it* (*card*). Antes de su implementación, estas historias se complementan con conversaciones adicionales con los usuarios y la definición de criterios de validación que confirman si se han cumplido los objetivos planteados.

Información en una Historia de Usuario

Es recomendable evitar formatos rígidos al definir historias de usuario y, en su lugar, adaptarlos a las características específicas de la empresa y del proyecto.

Además de los cuatro campos considerados esenciales, se pueden incluir otros que sean útiles para el desarrollo del proyecto.

Los campos esenciales son los siguientes:

- **ID:** Identificador único de la historia de usuario, correspondiente a una funcionalidad o tarea específica.
- **Descripción:** Aunque el estilo puede ser libre, debe responder a tres preguntas clave: ¿quién se beneficia?, ¿qué se desea lograr? y ¿cuál es el beneficio esperado? Se recomienda seguir el siguiente patrón:

  "Como [rol del usuario], quiero [objetivo], para poder [beneficio]."
- **Estimación:** Indica el esfuerzo necesario en términos de tiempo ideal para implementar la historia de usuario. Se pueden utilizar puntos de historia, que representan una medida teórica del tiempo de desarrollo por persona.
- **Prioridad:** Este campo permite establecer el orden en que deben ser implementadas las historias de usuario.

Dependiendo del tipo de proyecto, la dinámica del equipo y la organización, pueden ser útiles otros campos, como:

- Titulo
- Criterio de Validación: pruebas de aceptación acordadas con el cliente o usuario
- Valor: normalmente numérico que junto con otros elementos, ayuda en la priorización de las historias.
- Dependencias

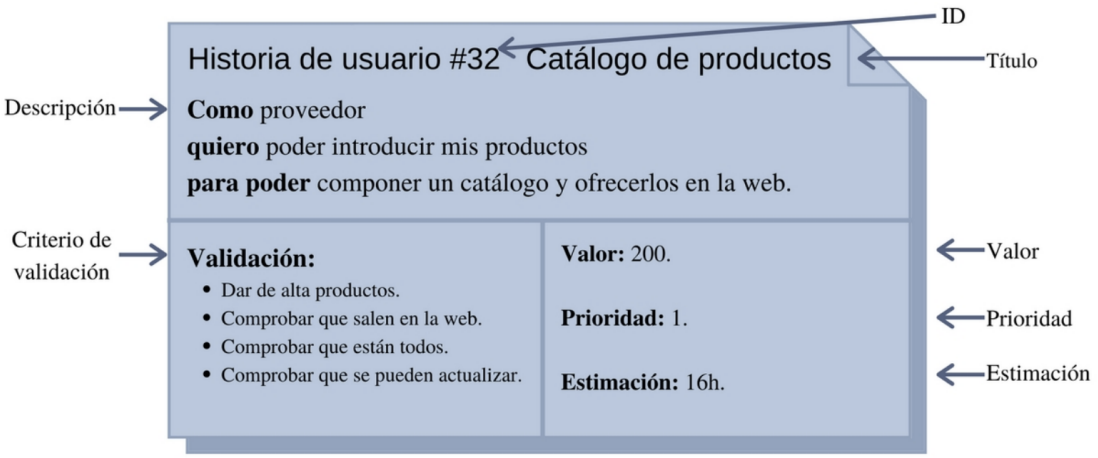

Figura 4: Ejemplo de una tarjeta de historia de usuario [5]

7.2. Epics, Temas y Tareas

Epics

Un ***epic*** es una superhistoria de usuario caracterizada por su gran tamaño y alta granularidad, en contraste con las historias de usuario que tienen una baja granularidad. Esta etiqueta se aplica a historias que requieren un esfuerzo significativo y que no pueden completarse en una sola vez o dentro de un solo *sprint*.

Los *epics* suelen tener un flujo asociado que permite su descomposición en historias de usuario; es decir, las historias resultantes están estrechamente relacionadas entre sí. A medida que un *epic* aumenta en prioridad y se aproxima a su implementación, el equipo lo descompone en historias de usuario de un tamaño más manejable, facilitando la aplicación de los principios y técnicas ágiles, como la estimación y el seguimiento cercano (normalmente diario).

Temas

Por encima de los *epics* se encuentran los **temas**, que representan una colección de *epics* y/o historias de usuario relacionadas. Los temas sirven para describir un sistema o subsistema en su totalidad. Por ejemplo, en un sistema de software de gestión contable, se podrían agrupar los *epics*: "Altas, bajas y mantenimiento de clientes", "Facturaciones puntuales y recurrentes", "Consultas de navegación y acciones de fidelización", "Pedidos" y "Devoluciones" bajo el tema de la gestión de clientes.

Tareas

Por debajo de las historias de usuario están las **tareas**. Estas son el resultado de la descomposición de las historias de usuario en unidades de trabajo más pequeñas y adecuadas para gestionar y seguir el avance de su ejecución.

Pila del Producto en Scrum

En Scrum y en metodologías ágiles en general, la pila del producto puede contener tanto historias de usuario como *epics*. Esta pila debe estar ordenada por prioridad, y el nivel de detalle de cada elemento debe corresponder a su posición en la lista.

Para elementos de baja prioridad que se encuentran al final de la lista, no tiene sentido mantener un alto nivel de detalle, ya que es probable que cambien a lo largo

del proyecto o incluso que no se desarrollen en absoluto.

A medida que nos acercamos a los elementos más prioritarios, el detalle debe aumentar. Por lo tanto, las historias de usuario deben encabezar la lista, mientras que los *epics* ocuparán posiciones más bajas, siendo así una representación clara de la relación entre prioridades y niveles de detalle.

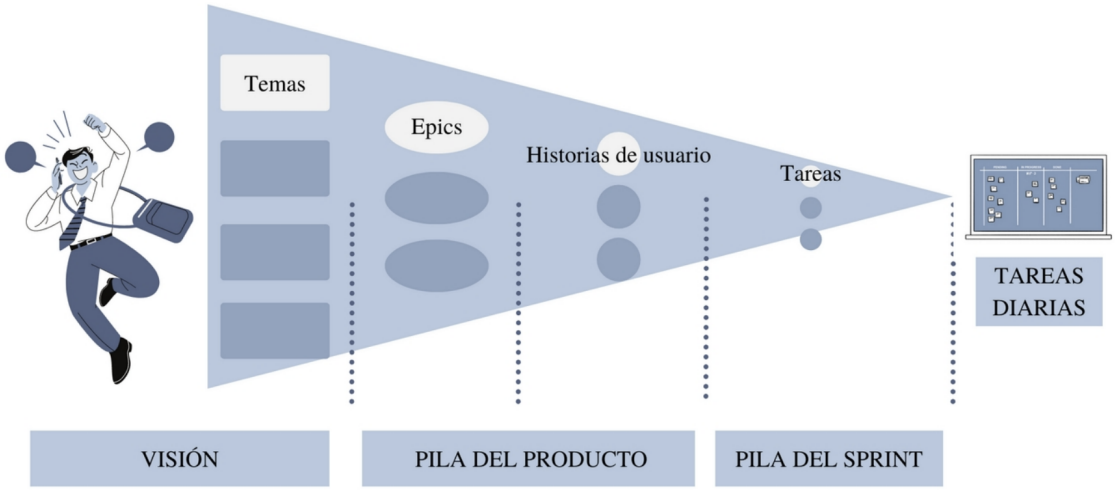

Figura 5: Gráfico con los cuatro niveles de tamaño con que trata los requisitos la gestión ágil [5]

Referencias

[1] Irene T. Luque Ruiz, Apuntes de la Asignatura Ingeniería del Software (2013).

[2] I. Sommervile, Ingeniería del Software, 9th Edition, Pearson, 2011.

[3] IEEE Recommended Practice for Software Requirements Specifications, IEEE Std 830-1998 (1998) 1–40doi:10.1109/IEEESTD.1998.88286.

[4] Amador Durán Toro y Bernárdez Jiménez, Metodología para la Elicitación de Requisitos de Sistemas Software, 2000.

[5] Alexander Menzinsky, Gertrudis López, Juan Palacio, Miguel Ángel Sobrino, Rubén Álvarez y Verónica Rivas, Historias de Usuario: Ingeniería de Requisitos Ágil, 2022.