"Un análisis de enfoques de Validación de Requerimientos"

Alumna: Sonia Raquel Santana
Ing. en Sistemas de Información
Estudiante de Maestría en Ingeniería de Software – UNLP
sonia.santana@uner.edu.ar

Director: Dr. Leandro Antonelli
Facultad de Informática – UNLP
lanto@lifia.info.unlp.edu.ar

Codirector: Mg. Pablo Thomas
Facultad de Informática – UNLP
pthomas@lidi.info.unlp.edu.ar

Trabajo Final presentado para obtener el grado de
Especialista en Ingeniería de Software

Facultad de Informática - Universidad Nacional
de La Plata

Mayo - 2022

Contenido

Capítulo 1 Introducción

En la industria del software el objetivo fundamental que se debería alcanzar es obtener productos software de calidad aplicando diferentes métodos y modelos bajo procesos estandarizados en el desarrollo de software, combinando técnicas, estrategias y todos aquellos aspectos que contribuyan a la búsqueda de conseguir calidad en los productos intangibles.

En la actualidad, se observa que los sistemas de software son cada vez más complejos, por lo que las exigencias de calidad en el desarrollo deben ser mayores. Es necesario que el desarrollo de software sea más riguroso para obtener un producto de adecuada calidad. La identificación y tratamiento de errores en etapas tempranas del proyecto de software es crucial para disminuir los costos de operación y evitar errores en etapas posteriores del desarrollo.

"La parte más difícil de desarrollar un sistema de software es decidir exactamente qué construir. Ninguna otra parte del trabajo conceptual es tan difícil como establecer los requerimientos técnicos detallados... Ninguna otra parte del trabajo paraliza tanto el sistema resultante si se hace mal. Ninguna otra parte es tan difícil de rectificar posteriormente" [1].

Según Brooks [1], los requerimientos son de vital importancia en un proyecto de desarrollo de software, ya que se convierten en el elemento fundamental que permite entender al cliente, analizar sus necesidades, validarlas, y gestionarlas conforme avanza el desarrollo del proyecto.

Según la IEEE [2] el área de conocimiento de requerimientos de software se ocupa de la obtención, análisis, especificación y validación de requerimientos de software así como la gestión de requerimientos durante todo el ciclo de vida del producto de software. Los requerimientos de software expresan las necesidades y restricciones impuestas a un producto de software que contribuyen a la solución de algún problema del mundo real.

En el marco de la Ingeniería de Requerimientos (RE por sus siglas en inglés Requirements Engineering) la validación de los requerimientos es una tarea fundamental en cualquier proyecto de Ingeniería de Software y debe ser un proceso continuo en el ciclo

de vida del desarrollo del sistema. El principal objetivo de la validación de requerimientos es confirmar que los requerimientos especificados sean representaciones de las necesidades y expectativas de los usuarios y que además sean completos, correctos y consistentes entre otras características.

Trabajar en la validación de requerimientos se está convirtiendo en un desafío para los equipos, clientes y usuarios. Existen diferentes causas que imponen problemas de comunicación, control, intercambio de conocimientos, confianza y retrasos en el desarrollo del software.

A partir de los resultados obtenidos en [4] y [5], presentados como artículos en una conferencia y un journal, se identificaron las siguientes contribuciones al proceso de Validación de Requerimientos:

- Funciones, componentes, entornos y características.
- Actividades de planificación.
- Técnicas de control e indicadores de rendimiento.
- Definición de estándares.
- Aceptación del cliente/usuario.
- Dominios de aplicación.
- Participación cliente/usuario.
- Etapas del ciclo de vida del desarrollo del software donde validan los requerimientos.

Además, se obtuvieron diferentes características en términos de recomendaciones de buenas prácticas para el desarrollo de enfoques de validación de requerimientos en cuanto a:

- Definición
- Modelado
- Operación
- Control
- Gestión de defectos
- Aceptación
- Soporte
- Dominio de aplicación

Gottesdiener [3] aporta una guía de referencia para facilitar la comunicación entre los equipos comerciales y técnicos a medida que se definen los requerimientos para los proyectos de software. Proporciona herramientas, técnicas y modelos a los miembros del equipo que tienen la necesidad de eliminar las barreras en la comunicación y ayudarlos a lograr sus objetivos. Según Gottesdiener, para analizar los requerimientos complejos se puede utilizar un modelo de requerimientos representado a través de una tabla donde se muestre las respuestas a las "4W's + H" (Who? What? When? Why? + How?) - (¿Quién? ¿Qué? ¿Cuándo? ¿Por qué? + ¿Cómo?).

Este trabajo tiene como objetivo seguir en la misma línea de investigación de [4] y [5] donde se busca identificar información sobre las características asociadas al proceso de validación de requerimientos en el ciclo de vida del software: la naturaleza de la información Quién, Qué, Cuando, Por qué y Cómo validar los requerimientos. La identificación de información de los enfoques se realiza a través de una revisión bibliográfica y una evaluación de diversos enfoques de validación de requerimientos para obtener sus características, necesidades de información y restricciones.

Este trabajo se organiza del siguiente modo: en el capítulo 2 se analizan los procesos de la Ingeniería de Requerimientos en el contexto de los modelos de desarrollo de software. Luego en el capítulo 3 se analiza el proceso de validación de requerimientos presentando distintas propiedades deseables en el modelo de requerimientos, técnicas y roles en la validación de requerimientos. En el capítulo 4 se realiza una revisión de la literatura para identificar los distintos enfoques de las prácticas relacionadas con el proceso de validación de requerimientos. Luego en el capítulo 5 se realiza un análisis que consiste en obtener información de los enfoques sobre las características asociadas al proceso de validación de requerimientos en el ciclo de vida del software: la naturaleza de la información. Además, se presentan los resultados del análisis comparativo realizado a los enfoques, donde se identificaron las fortalezas y debilidades de los trabajos en el proceso de Validación de Requerimientos. Finalmente se presentan las conclusiones y trabajo futuro.

Capítulo 2 Procesos de la Ingeniería de Requerimientos

Según Loucopoulos, la RE se describe más fácilmente por sus productos que por sus procesos. Sin embargo, es necesario contar con un punto de referencia como base para entender, evaluar y comparar diferentes propuestas en el área de la RE [6].

Se puede construir un marco para describir los procesos de RE considerando tres preocupaciones fundamentales de la RE:

- la preocupación por comprender un problema ("cuál es el problema")
- la preocupación por describir formalmente un problema
- la preocupación por alcanzar un acuerdo sobre la naturaleza del problema.

Cada una de las preocupaciones anteriores implica que deben llevarse a cabo algunas actividades para dar respuestas, y al hacerlo deben utilizarse algunos recursos. Por ejemplo, para entender un problema, el solucionador de problemas debe disponer de información relevante sobre el mismo (un recurso). A su vez, si esa información no está ya a disposición del solucionador de problemas debe obtenerse (una actividad). Del mismo modo, la información relevante debe ser validada para garantizar su exactitud, coherencia y relevancia (otra actividad) [6].

El proceso de RE puede verse como un conjunto de actividades que contienen una estructura en cada actividad. Loucopoulos plantea tres aspectos fundamentales para comprender, describir y acordar un problema en el proceso de RE: elicitación, especificación y validación. Se presentan como un proceso iterativo que pasa de una actividad a otra en todos los sentidos, desde el usuario al dominio del problema, como se muestra en la Figura 1 [6].

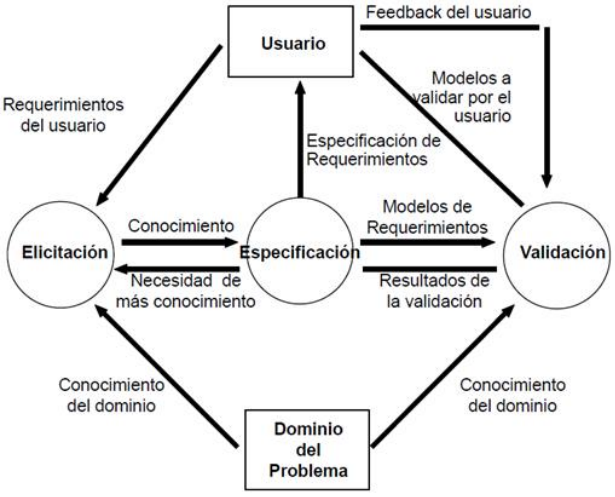

Fig. 1. Procesos RE según Loucopoulos.

El resto de este capítulo presenta cada uno los procesos de Loucopoulos y finalmente describe los ciclos de vida más significados para dar un marco a los procesos de requerimientos.

2.1 Elicitación de requerimientos.

En la mayoría de los casos, al inicio de un proyecto de software, un analista sabe muy poco sobre el problema de software que debe resolver. La única manera de revertir esta situación es profundizar en todo lo que es relevante para el problema y, en cierto sentido, convertirse en propietario del problema. Sin embargo, los problemas relacionados con el software suelen ser lo suficientemente complejos como para que el conocimiento relevante sobre ellos esté distribuido entre muchas personas, lugares y fuentes. Además, el conocimiento suele estar disponible en una variedad de anotaciones que van desde bocetos, a través de la prosa del lenguaje natural a modelos formales (por ejemplo, modelos matemáticos [6].

El propósito de la elicitación de requerimientos es obtener los conocimientos pertinentes al problema, que son utilizados para producir una especificación formal del desarrollo del software necesaria para resolver el problema. El analista debe convertirse en un experto en el dominio del problema. Las fuentes de conocimiento durante la fase de elicitación de requerimientos incluyen documentación, participantes del sistema, sistemas de software

existentes en ese dominio, sistemas similares en otros dominios y normas nacionales e internacionales, que limitan el desarrollo de software en ese dominio [6].

Las posibles fuentes del conocimiento del dominio pueden ser:

- expertos en la materia
- literatura sobre el dominio
- los sistemas de software existentes en ese ámbito
- aplicaciones informáticas similares en otros ámbitos
- normas nacionales e internacionales que limitan el desarrollo de software en ese ámbito
- otras partes interesadas en el sistema más amplio (por ejemplo, una organización) que alojará el sistema de software.

En la obtención de requisitos, el analista se enfrenta a las tareas de:

- identificar todas las fuentes de conocimiento de los requerimientos
- adquirir los conocimientos
- decidir sobre la relevancia de los conocimientos para el problema en cuestión
- comprender la importancia de los conocimientos obtenidos y su impacto en los requerimientos del software.

En la actualidad existe una variedad de técnicas para la obtención de requerimientos, cada una tiene puntos fuertes y débiles y se aplican a distintos tipos de problemas. Las técnicas más utilizadas para la elicitación de los requerimientos son las entrevistas y la creación de prototipos.

La elicitacion de conocimientos es un proceso que exige mucho trabajo y que requiere una gran cantidad de tiempo y recursos para el desarrollo de software. Esto se debe, en parte, a que la obtención de conocimientos (especialmente de los humanos) es una tarea intrínsecamente difícil. La mayoría de los métodos practicados no prescriben un resultado formal (modelo) para el proceso de elicitación, ya que tradicionalmente se cree que el único resultado formal de la RE debe ser el modelo de especificación de requerimientos.

Sin embargo, la experiencia demuestra que una mejor visión de la RE es la de un proceso de creación de modelos. Según este punto de vista, se crea una sucesión de modelos a lo largo de la RE, empezando por los modelos conceptuales y terminando con el modelo de

especificación de requerimientos. El analista comienza a formular modelos del dominio del problema en las primeras etapas de la RE. Estos modelos mentales que contienen el conocimiento dependiente del dominio, como los factores del entorno, los objetivos del dominio, las políticas, las restricciones, entre otros, suelen formularse y existir en la cabeza del analista. A medida que aumenta la comprensión del analista sobre el dominio del problema, estos modelos se vuelven más refinados y elaborados.

La elicitación se puede considerar como un proceso continuo de RE. Proporciona la 'materia prima' a otros procesos como la especificación, necesarios para la producción de un modelo formal. A este respecto, la elicitación se produce en paralelo con los procesos de especificación y validación [6].

2.2 Especificación de requerimientos.

La especificación puede ser vista como un contrato entre los usuarios y los desarrolladores de software que define el comportamiento funcional deseado del software y propiedades tales como el rendimiento y la fiabilidad entre otras, sin mostrar cómo se va a lograr dicha funcionalidad.

El proceso de especificación de requerimientos deriva modelos de requerimientos formales de software para ser utilizados en las etapas posteriores del desarrollo. El propósito de producir un modelo de especificación formal es doble [6]:

- El modelo de especificación se usa como un acuerdo entre los desarrolladores de software y los usuarios sobre lo que constituye el problema que debe ser resuelto por el sistema de software.
- El modelo de especificación es también un modelo para el desarrollo del sistema de software.

El proceso de especificación requiere como entrada conocimientos sobre el dominio del problema. Este conocimiento lo proporciona el proceso de elicitación. En la mayoría de los casos, el conocimiento de entrada viene en un formato "crudo" que debeconvertirse en información significativa para producir un modelo de especificación formal. Los conocimientos, por ejemplo, sobre las políticas generales de una organización deben interpretarse en relación con la forma en que afectan a los requerimientos de los sistemas de software. También se emplean en la especificación otros tipos de conocimientos

producidos por el proceso de validación. Este conocimiento establece lo que es válido y lo que no en la especificación formal, y como tal actúa como una fuerza para cambiar el modelo de requerimientos formales.

Es importante ver la especificación como un proceso complejo que requiere la retroalimentación del analista para el usuario y viceversa. El proceso utiliza y produce una variedad de modelos, incluyendo la especificación formal de los requerimientos. El proceso es *analítico* porque todos los diferentes tipos de conocimiento que el analista extrae del dominio del problema deben ser examinados y relacionados entre sí. La especificación también es *sintética* porque la heterogeneidad del conocimiento debe combinarse para producir un todo lógico y coherente que es la especificación de requerimientos [6].

La mayoría de los enfoques de la RE asumen que el resultado de este proceso es el modelo de especificación de requerimientos. Sin embargo, es más apropiado ver el proceso de especificación como la producción de una variedad de modelos que corresponden a diferentes puntos de vista del problema. En este sentido, la especificación de requerimientos produce [6]:

- Modelos orientados al usuario que especifican el comportamiento, las características no funcionales, etc. del software y que sirven como punto de entendimiento entre el analista, el cliente y el usuario
- Modelos orientados al desarrollador que especifican las propiedades funcionales y no funcionales del sistema de software, así como las limitaciones de recursos, las restricciones de diseño, etc., que actúan como planos para las etapas posteriores de desarrollo.

Todos los modelos anteriores corresponden a modelos empresariales, modelos de requerimientos funcionales y modelos de requerimientos no funcionales. Sin embargo, algunos métodos de requerimientos no distinguen entre los diferentes modelos por razones de simplicidad y esfuerzo. Estos métodos, por ejemplo, utilizan el mismo modelo de especificación funcional para todas las clases de requerimientos. Sin embargo, esto no siempre es apropiado, ya que una notación de especificación que es perfectamente clara para los desarrolladores puede ser difícil de comprender para los usuarios.

La especificación de requerimientos es el proceso central de la RE. La especificación controla los procesos de elicitación y validación de la siguiente manera. Durante la

especificación puede resultar evidente que se necesita más información sobre el problema. Esto desencadenará el proceso de elicitación que, a su vez, proporcionará la información necesaria. Por otra parte, algún cambio en el ámbito del problema (por ejemplo, un cambio en alguna suposición sobre el ámbito) puede provocar un cambio en el modelo de especificación. Por lo tanto, la elicitación puede tener lugar durante el proceso de especificación. Entre la especificación y la validación se producen interacciones similares. La finalización de alguna parte del modelo de especificación puede provocar la necesidad de validación [6].

2.3 Validación de requerimientos.

Cualquier modelo de requerimientos (formal o informal) está sujeto a la validación y, por tanto, proporciona una entrada al proceso. Por ejemplo, los conocimientos sobre el dominio del problema deben ser validados, es decir, se comprueba su exactitud, coherencia, relevancia para el proyecto, etc. Del mismo modo, alguna parte del modelo de requerimientos formales debe validarse por partes y en su conjunto [6].

La validación es un proceso que requiere interacciones entre los analistas, los clientes del sistema previsto y los usuarios del ámbito del problema. Es similar al proceso científico de formular una nueva teoría (especificación) y posteriormente probarla mediante la realización de experimentos (validación). La validación proporciona un modelo de requerimientos coherente y acorde con las expectativas de los usuarios. Esto no significa que el modelo sea correcto en ningún sentido. En la mayoría de los casos, la validación supone un compromiso entre lo que desean los usuarios y lo que es posible y factible con las limitaciones del proyecto.

La validación está siempre presente en todas las etapas de la RE. La necesidad de validación se desencadena por la adquisición de nuevos conocimientos sobre el dominio del problema (elicitación), o por la formulación de un modelo de requerimientos (especificación). La validación también es necesaria durante las fases de análisis y síntesis de la especificación de requerimientos, ya que hay que comprobar lacorrección de la información analizada y la coherencia y consistencia lógica de los requerimientos sintetizados [6].

La validación de requerimientos es un proceso continuo de RE que tiene como objetivo garantizar que se está abordando el problema correcto en cualquier momento. La validación de requerimientos se define como el proceso que certifica que los requerimientos del modelo son consistentes con las necesidades de los clientes y usuarios. Esta visión de la validación es más general que los encontrados en la literatura porque trata la validación como un proceso continuo que procede en paralelo con la elicitación y la especificación [6].

Los procesos generan productos que a su vez son alimentados por otros productos. Los productos son las interfaces entre los procesos. Por lo tanto, los productos y procesos requieren ser gestionados a través de actividades de:

- Planificación.
- Administración.
- Control.
- Organización.

Esto constituye la Gestión de Requerimientos que interactúa con todos los procesos de RE.

Es importante, tanto por razones conceptuales como prácticas, hacer la distinción entre validación y otros procesos y actividades de RE como se argumenta en esta Sección y de nuevo en el Capítulo 3.

2.4 Modelos de desarrollo de software

En esta sección se analizan los procesos de RE, en el contexto de los siguientes modelos de desarrollo de software [6]:

- el modelo de cascada
- el modelo en espiral
- el modelo de creación de prototipos
- el modelo operativo
- el modelo transformacional
- el modelo basado en el conocimiento
- el modelo de análisis de dominio

Modelo de cascada

La filosofía que describe el "modelo de cascada" [7] establece que el desarrollo de software consiste en una transformación por etapas desde el dominio del problema hasta la solución a través de una serie de fases que se satisfacen en su totalidad antes de que comiencen sus sucesoras. Según esta filosofía, la RE es la fase en la que se adquieren, analizan y validan los requerimientos del software y se elabora una especificación formal de los mismos. Esta fase suele ir precedida de otra conocida como *Análisis de Mercado* que define el contexto para la fase de RE. Por último, a la RE le sigue la fase de diseño, que se ocupa de especificar la solución de software a la especificación de requerimientos. El modelo de cascada considera la RE como una fase de *comprensión* a la que suceden las fases de *invención* (diseño) y *realización* (codificación).

En la literatura, el enfoque en cascada ha sido criticado por una serie de razones, como [6]:

- falta de participación del usuario en el desarrollo una vez finalizada la especificación de requerimientos [8]
- inflexibilidad para acomodar los prototipos [9]
- separación irreal de la especificación del diseño [10]
- incapacidad para acomodar el software reutilizado [11]
- problemas de mantenibilidad [12]
- integración de sistemas complicados [13]

Modelo espiral

El modelo espiral de desarrollo de software reconoce la naturaleza iterativa del desarrollo y la necesidad de planificar y evaluar los riesgos en cada iteración. Según este modelo, en cada una de las fases de desarrollo del software deben realizarse las siguientes actividades [14]:

- planificar las próximas fases
- determinar los objetivos, las alternativas y las limitaciones
- evaluar las alternativas, identificar y resolver los riesgos
- desarrollar, verificar el siguiente nivel del producto.

El modelo en espiral introduce los subprocesos adicionales de la RE, conocidos como *análisis de riesgos de los requerimientos* mediante técnicas como *la simulación* y la *creación de prototipos* y la *planificación del diseño*. Estas adiciones tienen como objetivo reducir el riesgo de cambio en alguna etapa posterior. Al evaluar la viabilidad de los requerimientos propuestos, por ejemplo, el enfoque reduce el riesgo de tener que repetir RE una vez que se ha llegado a una etapa (por ejemplo, el diseño) en la que podría descubrirse que no es factible producir el sistema de software requerido. Sin embargo, el modelo en espiral no puede hacer frente a cambios imprevistos durante alguna fase del desarrollo y su impacto en otras fases. Si, por ejemplo, se produce un nuevo objetivo empresarial mientras el desarrollo ha llegado a la fase de codificación, entonces es inevitable tener que repetir RE y el diseño [6].

Modelo de creación de prototipos

En el contexto del desarrollo de software, la *creación de prototipos* es una técnica que construye y experimenta con una versión maqueta del sistema de software, con el fin de obtener una cierta comprensión de la funcionalidad y el comportamiento quese requiere de él [15]. Hay que destacar que la creación de prototipos es una técnica ampliamente aceptada que puede utilizarse en el contexto de cualquier modelo de desarrollo de software.

Según el modelo de desarrollo de prototipos, los procesos de la RE, como la obtención, el análisis y la especificación se ocupan de la planificación, el desarrollo y la experimentación de un prototipo. Tras una comprensión inicial (posiblemente incorrecta y/o incompleta) de las necesidades de los usuarios, los desarrolladores de prototipos determinan los objetivos y el alcance del prototipo [6].

Tras una serie de revisiones necesarias, los desarrolladores suelen considerar que el prototipo satisface todos los objetivos. Llegados a este punto, los desarrolladores tienen dos opciones:

- perfeccionar el prototipo hasta convertirlo en un sistema completo o
- iniciar un proceso de desarrollo convencional tras haber desarrollado el prototipo.

Si se elige la primera opción, puede decirse que el prototipo es la especificación de requerimientos del sistema. En caso de que los desarrolladores prefieran la segunda opción,

habrá que desarrollar una especificación de requerimientos formal basada en la mejor comprensión de los requerimientos, obtenida en el proceso de creación de prototipos.

Modelo de especificación operativa

Una especificación operativa es un modelo de sistema que puede evaluarse o ejecutarse para generar el comportamiento del sistema de software. La especificación operativa se construye en una fase temprana de la RE y su objetivo es analizar no sólo el comportamiento requerido, sino también la estructura necesaria del sistema de software. Los defensores de este enfoque afirman que es imposible separar la estructura (el "cómo") del comportamiento (el "qué") cuando se analiza un sistema [16].

El enfoque operativo entrelaza deliberadamente las consideraciones de "qué" y "cómo" en un modelo de especificación ejecutable. De este modo, el enfoque intenta garantizar una validación temprana del modelo de requerimientos (ejecutable) por parte del usuario, así como la viabilidad de la solución propuesta. El enfoque operativo considera los procesos de RE, como la obtención, la especificación y la validación, de la siguiente manera [6]:

- el proceso de elicitación se lleva a cabo en una fase inicial, antes de la construcción de la especificación operativa,
- el proceso de especificación coincide con la construcción de un modelo de especificación ejecutable,
- el proceso de validación coincide con el ejercicio del modelo de especificación operativa.

Modelo de transformación

El enfoque transformacional del desarrollo de software intenta automatizar las etapas de desarrollo que requieren mucho trabajo, como el diseño y la implementación, utilizando el concepto de transformación. Una transformación se define como un mapeo de un objeto más abstracto (como una especificación) a otro menos abstracto (como un diseño o un trozo de código). El enfoque transformacional aboga por el uso de una serie de transformaciones que convierten una especificación en un sistema de software concreto [17]. El enfoque transformacional implica la necesidad de una especificación formal como entrada inicial, ya que las transformaciones automáticas sólo pueden aplicarse a un objeto

formal (especificación) [6].

El código alterado se produce mediante un proceso de dos pasos. En primer lugar, se introducen cambios en la serie de transformaciones que produjeron el código original que se registraron durante el desarrollo. En segundo lugar, el conjunto modificado de transformaciones se "reproduce" para obtener la nueva versión del sistema de software.

Las etapas del modelo de transformación se corresponden con los procesos de RE de la siguiente manera:

- la obtención de los requerimientos es la fase en la que se obtiene un modelo inicial de requerimientos informal e incompleto, antes de comenzar el desarrollo transformacional,
- la especificación de requerimientos es la fase que produce el modelo de especificación formal,
- la validación de requerimientos es la fase de transformación en la que el modelo formal es validado por el usuario.

El enfoque transformacional también ha sido criticado por su necesidad de crear y validar un modelo de especificaciones formal, así como por la dificultad de automatizar el proceso de transformación.

Modelo basado en el conocimiento

Los enfoques basados en el conocimiento de la RE que se caracterizan por el uso de herramientas de software inteligentes para realizar o apoyar alguna actividad dentro de la RE. El término "inteligente" implica que las herramientas incorporan una base de conocimientos que consiste en:

- conocimientos sobre cómo realizar algún aspecto de la RE (por ejemplo, elicitación, especificación, validación) y/o,
- conocimiento sobre las características de algún dominio del problema (llamado conocimiento del dominio) que puede emplearse en la RE.

El paradigma basado en el conocimiento no implica necesariamente el uso de un modelo de desarrollo de software diferente, pueden adoptar cualquiera de los modelos de cascada, prototipo, operativo, entre otros. Por lo tanto, las principales diferencias entre los enfoques basados en el conocimiento y los no basados en el conocimiento existen en el

grado de uso de herramientas inteligentes en un proceso dentro de la RE. Por ejemplo, la validación de requerimientos se realiza convencionalmente dejando que el usuario inspeccione el modelo de requerimientos (que puede ser un texto, diagramas, prototipo, entre otros.). Sin embargo, si la validación se realiza comprobando la coherencia del modelo de requerimientos con las reglas (que establecen Cuando un modelo es coherente) almacenadas en una base de conocimientos, la validación se convierte en un enfoque basado en el conocimiento [6].

Modelo de análisis de dominio

El análisis de dominio se ha considerado una actividad que tiene lugar antes de la RE [18]. Mientras que la RE se ocupa de analizar y especificar el problema del desarrollo de una aplicación de software, el análisis de dominio se ocupa de identificar los puntos comunes entre diferentes aplicaciones del mismo dominio. El resultado del análisis de dominios es un conjunto de objetos, relaciones y reglas que son comunes en un dominio de problemas que pueden reutilizarse en diferentes aplicaciones. Por ejemplo, las aplicaciones de software que se ocupan de las reservas de billetes de avión consideran un conjunto estándar de objetos como pasajero, vuelo, reserva, billete, entre otros. El análisis de dominios sugiere que conceptos como los mencionados pueden abstraerse y organizarse en bibliotecas para que puedan reutilizarse "sin más" en futuras aplicaciones. En este sentido, el análisis de dominios cambia radicalmente nuestra forma de entender la RE por las siguientes razones [6]:

- fases como la comprensión del problema que tradicionalmente se consideran en la RE se reducen a "seleccionar y recuperar el contenido de la biblioteca apropiada que contiene los resultados del análisis del dominio considerado,
- el esfuerzo para la obtención de requerimientos se reduce significativamente debido a que la obtención ya se ha realizado como parte del análisis de dominio,
- la especificación de los requerimientos consiste en la selección de un componente adecuado de la biblioteca de componentes de análisis reutilizables, con una posible adaptación si es necesario,
- por último, la necesidad de validación se reduce, debido a que los componentes de la biblioteca ya han sido validados como parte del análisis del dominio.

Se puede decir que todos los modelos de desarrollo de software tienen en cuenta los principales procesos de la RE, a saber, la obtención, la especificación y la validación. También es cierto que los diferentes enfoques tienden a infravalorar o sobrevalorar uno o más procesos de la RE. Tradicionalmente, el proceso menos enfatizado y considerado de la RE ha sido la validación. Asimismo, los últimos enfoques de la RE intentan rectificar el hecho de que el proceso siguesiendo el más laborioso del desarrollo de software [6].

Capítulo 3 Validación de Requerimientos

En el marco de la RE la validación de los requerimientos es una tarea fundamental en cualquier proyecto de Ingeniería de Software y debe ser un proceso continuo en el ciclo de vida del desarrollo del sistema. El principal objetivo de la validación de los requerimientos es confirmar que los requerimientos especificados sean representaciones de las necesidades y expectativas de los usuarios [19] [20] [21] y deben ser completos, correctos y consistentes [21] entre otras características.

Pressman [22], considera que la calidad de los productos del trabajo que se generan como consecuencia de la ingeniería de los requerimientos se evalúa durante el paso de validación. La validación en ese contexto permite garantizar que todos ellos han sido enunciados sin ambigüedades; que se detectaron y corrigieron las inconsistencias, las omisiones y los errores, y que los productos del trabajo se presentan conforme a los estándares establecidos para el proceso, el proyecto y el producto.

Pressman [22], señala que a medida que se crea cada elemento del modelo de requerimientos, se estudia para detectar inconsistencias, omisiones y ambigüedades. Los participantes asignan prioridades a los requerimientos representados por el modelo y se agrupan en paquetes de requerimientos que se implementarán como incrementos del software. La revisión del modelo de requerimientos aborda las preguntas siguientes:

- ¿Es coherente cada requerimiento con los objetivos generales del sistema o producto?
- ¿Se han especificado todos los requerimientos en el nivel apropiado de abstracción? Es decir, ¿algunos de ellos tienen un nivel de detalle técnico que resulta inapropiado en esta etapa?
- El requerimiento, ¿es realmente necesario o representa una característica agregada que tal vez no sea esencial para el objetivo del sistema?
- ¿Cada requerimiento está acotado y no es ambiguo?
- ¿Tiene atribución cada requerimiento? Es decir, ¿hay una fuente (por lo general una individual y específica) clara para cada requerimiento?
- ¿Hay requerimientos en conflicto con otros?
- ¿Cada requerimiento es asequible en el ambiente técnico que albergará el sistema o producto?

- Una vez implementado cada requerimiento, ¿puede someterse a prueba?
- El modelo de requerimientos, ¿refleja de manera apropiada la información, la función y el comportamiento del sistema que se va a construir?
- ¿Se ha "particionado" el modelo de requerimientos en forma que exponga información cada vez más detallada sobre el sistema?
- ¿Se ha usado el patrón de requerimientos para simplificar el modelo de éstos? ¿Se han validado todos los patrones de manera apropiada? ¿Son consistentes todos los patrones con los requerimientos del cliente?

Éstas y otras preguntas deben plantearse y responderse para garantizar que el modelo de requerimientos es una reflexión correcta sobre las necesidades del participante y que provee un fundamento sólido para el diseño.

Según Kotonya [23] la validación de requerimientos se refiere a verificar la coherencia, integridad y corrección del documento de requerimientos, y también se establece que los requerimientos deben comprobarse para que los mismos sean: válidos, comprensibles, consistentes, trazables, íntegros, reales y verificables [24]. Según Bahill [25], el proceso de validación de requerimientos consiste en primer lugar asegurar que un conjunto de requerimientos es: correcto, completo y consistente, en segundo lugar ver si se puede crear un modelo que cumpla con los requerimientos, y por último que se pueda construir y probar una solución de software en el mundo real para demostrar que cumple con los requerimientos de las partes interesadas.

En la Figura 2 se ilustra el proceso de validación de requerimientos, según Kotonya y Sommerville [24], las entradas del proceso son:

- los documentos de requerimientos,
- el conocimiento organizacional,
- y los estándares organizacionales son entradas del proceso.

Y los resultados del proceso son:

- la lista de problemas,
- y las acciones acordadas para resolver los problemas.

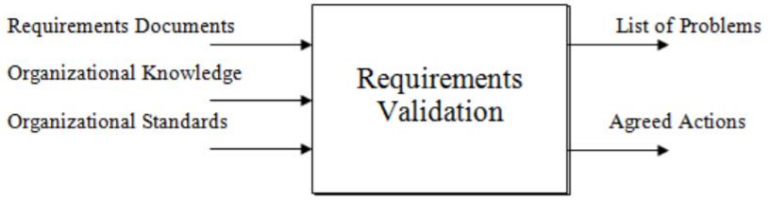

Fig. 2. Proceso de validación de requerimientos [24].

De acuerdo con la norma EIA632 [26], el proceso de validación de requerimientos debería garantizar el éxito del sistema a nivel de desarrollo e implementación asegurando que los requerimientos son necesarios y suficientes para crear soluciones de diseño apropiadas en un ciclo de vida a nivel de ingeniería o de empresa.

Trabajar en el contexto de la validación de requerimientos se está convirtiendo en un desafío para los equipos, clientes y usuarios, existen diferentes causas que imponen problemas de comunicación, control, intercambio de conocimientos, confianza y retrasos en el desarrollo del software [6].

La validación de requerimientos garantiza que los requerimientos sean necesarios y estén suficientemente especificados para satisfacer las necesidades del usuario. Las actividades de validación de requerimientos detectan y corrigen cualquier requerimiento innecesario e incorrecto [27].

La detección de un error en la etapa posterior del desarrollo de software suele llevar más tiempo y ese error es más costoso que si se detecta en la etapa inicial [28], [29], [30], [31]. Por ejemplo, Boehm y Basili [28] realizaron un estudio sobre la reducción de defectos en el software indican que:

- Encontrar y corregir defectos después del lanzamiento de un producto es a menudo 100 veces más costoso que corregir el mismo error en la fase de requerimientos y modelado. Esta medida puede variar dependiendo de la complejidad del sistema. Actividades como el análisis y diseño de requerimientos, involucramiento temprano, la verificación y validación en todo el ciclo de vida del software, prototipos y simulación permiten evitar costosas correcciones.
- Permite la reducción de costos: la relación del costo por el esfuerzo requerido para dar solución a los defectos encontrados en una determinada fase del ciclo de vida del software, se presenta en la Tabla 1, por ejemplo, si se detecta un defecto en la

fase de operación del sistema, su corrección presentaría un costo mayor al que habría sido si se encontraba en las fases de pruebas de sistema / integración o pruebas de aceptación [32].

Tabla 1. Proporción del costo por fase [32].

Fase donde se encuentra el defecto	Porción de costo
Requerimiento	1
Diseño	3-6
Codificación	10
Pruebas de sistema / Integración	15-40
Pruebas de aceptación de usuario	30-70
Operación	40-1000

- Proyectos de software gastan aproximadamente 40 a 50 por ciento de su esfuerzo en re trabajo evitable. El esfuerzo dedicado a la corrección de los defectos de software que podrían haber sido descubiertos, corregidos o evitado por completo en etapas tempranas a un menor costo [33].
- Aproximadamente el 80 por ciento del re trabajo evitable proviene del 20 por ciento de los defectos [33].
- Dos de las principales fuentes de re trabajo se originan por:
    - Requisitos especificados apresuradamente
    - Definición tardía de requisitos no nominales causan el mayor re trabajo en diseño, arquitectura, y en el código.
- Alrededor del 80 por ciento de los defectos provienen del 20 por ciento de los módulos, y aproximadamente la mitad de los módulos son defectos [33].
- Casi todos los defectos se agrupan en la mitad de los módulos. El reconocimiento de las características de los módulos propensos a errores en un entorno particular puede resultar útil.
- Acerca del 90% del tiempo de inactividad viene en su mayoría del 10 por ciento de los defectos [33].
- Algunos defectos afectan de manera desproporcionada el tiempo de inactividad y la fiabilidad de un sistema [33].

- La revisión de pares captura el 60 por ciento de los defectos. Estudios presentan que la revisión de pares ofrece una técnica efectiva debido que permite encontrar entre el 31 y el 93 por ciento de los defectos, con una mediana de alrededor de 60 por ciento [33].

Además, una encuesta realizada por Hofmann y Lehner muestra que los proyectos de software exitosos han invertido alrededor del 28% del esfuerzo total en la fase de requerimientos [34].

Una actividad importante y necesaria en el proceso de RE que puede evitar que los errores se propaguen a una fase de desarrollo posterior es la validación de requerimientos. La validación de requerimientos es una tarea difícil porque, por naturaleza, una especificación de requerimientos puede ser interpretada de diferentes maneras y en diferentes contextos por las partes interesadas.

Según Gottesdiener, la validación logra detectar errores e inconsistencias en los requerimientos para garantizar que los requerimientos representen adecuadamente las necesidades del usuario, para reducir los costos asociados con la corrección de los defectos de implementación que se originan en los requerimientos, y aumentar la calidad del software [3].

Según Kof, el 56 por ciento de los errores en los proyectos de software se deben a los errores generados en la fase de requerimientos, como se muestra en la Figura 3. La mitad de estos errores se deben a los requerimientos incompletos y ambiguos y la otra mitad se debe a los requerimientos que se omiten [35].

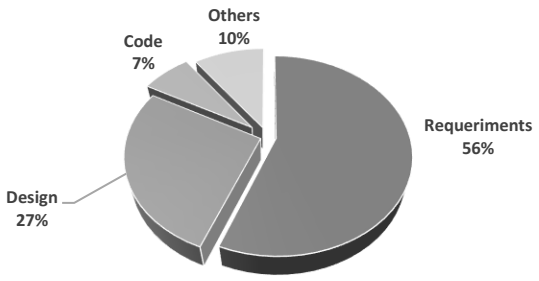

Fig. 3. Distribución de errores [35].

Si los requisitos están incompletos, los profesionales del software terminan construyendo un producto de software que no satisface todas las necesidades del cliente y

del usuario. Como se ilustra en la Figura 4, Noritaki Kano desarrolló un modelo de la relación entre la satisfacción del cliente y los requerimientos de calidad [36].

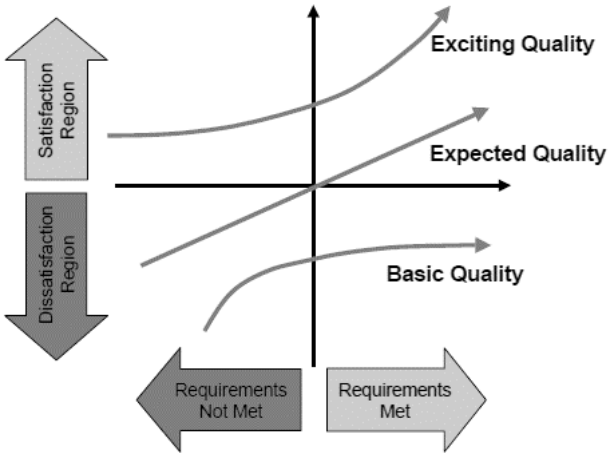

Fig. 4. Modelo de Kano para requerimientos de calidad [36].

La satisfacción del cliente aumenta a medida que se cumplen sus requerimientos explícitos. Cuando se cumplen los requerimientos explícitos, el cliente pasa de estar insatisfecho con el producto a ser un cliente satisfecho. Hay un nivel básico de requerimientos de calidad que un cliente espera que tenga el producto. Estos son requerimientos asumidos por el cliente y, por lo general, no se establecen explícitamente. Por ejemplo, esperan que un automóvil tenga cuatro cubiertas, un motor que funcione y un volante. Este nivel de requerimientos no satisface al cliente. Tenga en cuenta que toda la línea básica está en la región de insatisfacción. La ausencia de este nivel de requerimientos de calidad aumentará la insatisfacción del cliente. La calidad emocionante es el nivel de requerimientos innovadores y representa elementos inesperados. Los requerimientos de calidad esperados son los que los profesionales pueden obtener con facilidad si hablan con las partes interesadas del producto. Sin embargo, es fácil pasar por alto los requerimientos de calidad básicos y emocionantes si no se hace un trabajo de obtención y análisis detallado de los requerimientos. Además, si los profesionales pierden un grupo de partes interesadas o si no se logra involucrar a los usuarios en el proceso de requerimientos pueden terminar con brechas en sus requerimientos esperados [36].

3.1 Propiedades del modelo de requerimientos

La validación es una actividad "siempre presente" que tiene lugar cada vez que se adquiere, se analiza o se integra una parte de los requerimientos con el resto del modelo de requerimientos. La validación de requerimientos es también un proceso principalmente no estructurado. Esto significa que no es posible aplicar un procedimiento algorítmico que permita obtener un modelo de requerimientos validado. Sin embargo, hay un conjunto de tareas que se realizan durante la validación y que se aplican por igual a todas las metodologías y formalismos utilizados para construir el modelo de requerimientos. El objetivo de estas tareas es identificar la existencia de una serie de propiedades deseables en el modelo de requerimientos, a saber [6]:

Consistencia interna

Un modelo de requerimientos es internamente consistente si no se pueden derivar de él conclusiones contradictorias. Ante los problemas de comprobación de la coherencia, algunos investigadores han sugerido que un modelo de requerimientos se especifique utilizando la lógica matemática [37]. En lógica podemos ver los requerimientos como sentencias lógicas que pueden ser *verdaderas* o *falsas*. Las frases de requerimientos complejas pueden construirse a partir de otras más sencillas utilizando conectores lógicos como *y, o, no* e *implica*. Un ejemplo de requerimientos complejo formado a partir de dos simples es el siguiente:

"un libro está disponible para ser prestado"

"un libro está fuera de la biblioteca"

"un libro está disponible para ser prestado" *o* "un libro está fuera de la biblioteca".

Poder manipular los requerimientos de forma lógica cuando es posible, tiene una serie de ventajas. Un probador de teoremas es un programa que puede inferir nuevas sentencias lógicas combinando las existentes, utilizando reglas de inferencia. Un probador de teoremas aplicado a un modelo de requerimientos lógicos puede ayudar al analista a descubrir ciertos tipos de contradicciones [6].

No ambigüedad

La no ambigüedad significa que un requerimiento no puede interpretarse de más de una manera. Es fácil introducir ambigüedades en un modelo de requerimientos, especialmente cuando se utiliza el lenguaje natural. Por lo tanto, una de las tareas más importantes del analista es eliminar las ambigüedades del modelo de requerimiento utilizando su sentido común y las reglas de la lógica, aclarando las especificaciones ambiguas junto con los usuarios [6].

Consistencia externa

La coherencia externa es la concordancia entre lo establecido en el modelo de requerimientos y lo que es cierto en el dominio del problema. Por lo tanto, hay que garantizar que lo que se dice en el modelo de requerimientos está en consonancia con el dominio del problema. La mayor parte de los datos que aparecen en el modelo de requerimientos proceden de entrevistas con los usuarios o del estudio de la literatura sobre el ámbito del problema entre otros. Si los datos sobre el ámbito son volátiles (es decir, cambian con frecuencia, algo que suele ocurrir cuando, por ejemplo, el software se va a integrar en un nuevo hardware que aún no se ha desarrollado), habrá que hacer todo lo posible por mantener los datos del modelo de requerimientos actualizados [6].

Minimalidad

La minimización es una característica importante que debe estar presente en el documento de requerimientos. Lo contrario de la minimización se llama sobreespecificación, que es simplemente la tendencia a incluir en el modelo de requerimientos más de lo necesario. Normalmente, la sobreespecificación es un intento de prescribir una solución de diseño al mismo tiempo que especificamos el requerimiento. Sin embargo, esto es un error porque limita las posibles soluciones entre las que pueden elegir los diseñadores [6].

Completitud

Un modelo de requerimientos es completo cuando no omite información esencial sobre el ámbito del problema que podría dar lugar a un sistema que no satisfaga las necesidades del usuario. La exhaustividad es algo difícil de comprobar en un modelo de requerimientos, ya que no existe un sistema de procedimiento formal para realizarlo. Un modelo de

requerimientos contiene los objetivos que deben alcanzarse en un dominio del problema, así como las reglas, los hechos y las restricciones que se aplican al dominio y al sistema de software que operará en él. La falta de captura de cualquiera de estos ingredientes de un modelo de requerimientos tendrá un impacto en la corrección del modelo [6].

- La omisión de un objetivo, por ejemplo, dará lugar a un modelo que no representará todas las necesidades del usuario.
- El hecho de no capturar una regla o hecho del dominio modelado, impedirá que el sistema de software sea un modelo correcto de ese dominio.
- La ausencia de una restricción en el modelo de requerimientos puede conducir a un comportamiento incorrecto del sistema de software.

Los tipos de comprobación de integridad más difíciles, es decir, la comprobación de los objetivos, hechos o restricciones que faltan, pueden ser asistidos de varias maneras, tales como [6]:

- Al observar sistemas con requerimientos similares al que se está analizando, podemos identificar los requerimientos "olvidados" (objetivos).
- Al asumir una organización jerárquica de los requerimientos, podemos identificar los requerimientos que no tienen su correspondiente "nivel superior". Si falta la justificación para incluir algún requerimiento en particular, esta justificación que falta podría ser algún otro requerimiento olvidado.

Una buena forma general de comprobar la integridad del modelo de requerimientos es utilizar la creación de prototipos. Mediante la creación de prototipos del modelo de requerimientos, el analista y los usuarios participantes pueden encontrar omisiones evidentes (así como, por supuesto, incoherencias, ambigüedades entre otras).

Redundancia

Un requerimiento es redundante si también puede obtenerse de alguna otra parte del modelo de requerimientos, o si simplemente se trata de alguna propiedad o comportamiento no deseado en el sistema de software. Las definiciones múltiples de los requerimientos no son características deseables de un modelo de requerimientos, lo ideal es que un requerimiento sea identificable en un solo lugar del modelo de requerimientos. Sin

embargo, hay que tener cuidado cuando un requerimiento está implícito en otros requerimientos establecidos. Si un requerimiento está implícito (en un sentido lógico) a partir de otros requerimientos y, al mismo tiempo, se declara explícitamente, se aplica una de las siguientes condiciones [6]:

- la declaración explícita del requerimiento puede eliminarse, siempre que no se tema que el requerimiento implícito sea ignorado,
- el requerimiento explícito puede permanecer en el documento junto con una referencia a los requerimientos que lo implican. Si posteriormente se elimina alguno de esos requerimientos, el analista deberá volver a examinar la situación del requerimiento implícito.

Decidir si un requerimiento es redundante o no es una actividad que requiere comprender las expectativas de los usuarios del sistema (a veces llamada "lista de deseos"), la viabilidad de los requerimientos propuestos y también asignar prioridades a los requerimientos. Una forma de hacerlo es asignar a cada requerimiento un valor de una gama, empezando por "absolutamente esencial", pasando por "algo agradable de tener en el sistema", hasta llegar a "redundante" [6].

Corrección

Una especificación de requerimientos es correcta si y sólo si todo requerimiento contenido en ella debe describir de manera precisa la funcionalidad o restricción que se debe tener en cuenta para construir el software y este debe ser necesario para el software y no debe entrar en conflicto con otro requerimiento.

Verificabilidad

La especificación de requisitos es verificable si todos los requerimientos son no ambiguos y pueden ser cuantificados de manera que permita hacer uso de los métodos de verificación: inspección, análisis, demostración o pruebas.

Trazabilidad

La especificación de requerimientos es rastreable si y solo si para cada requerimiento contenido en ella se conoce su origen y puede referenciarse como origen en posteriores

documentos durante el ciclo de desarrollo del software, cada requerimiento puede rastrearse hacia atrás y hacia adelante.

Comprensibilidad

La especificación de requerimientos es comprensible si puede ser entendido por los usuarios y clientes del proyecto. Debe evitarse la utilización de términos con conceptos técnicos.

Legibilidad

La especificación de requerimientos es legible cuando el lenguaje en el que está escrita es fácil de entender, tanto por los clientes como los analistas. La legibilidad se extiende no solo a la especificación escrita, sino también a los formalismos gráficos como los diagramas de casos de uso.

Prioridad

En la especificación de requerimientos se establece el orden de implementación de una serie de requerimientos de software, tomando en consideración, algún conjunto definido de variables objetivo (criterios), como ser: satisfacción de los stakeholders, costos, necesidades del cronograma, entre otras.

Validez

La especificación de requerimientos es válida si y solo si cada requerimiento contenido en ella representa las necesidades y expectativas de los usuarios.

Modificabilidad

La especificación de requerimientos es modificable si y solo si su estructura y redacción permite que los cambios se puedan realizar fácil, completa y consistentemente.

En un mundo ideal, el proceso de validación debería durar tanto como se requiera. El modelo de requerimientos debe ser revisado por el analista junto con el usuario y herramientas de software cuando sea necesario para evitar omisiones, incoherencias,

violación de restricciones. El comportamiento del modelo debe exhibirse mediante técnicas como la creación de prototipos, la animación entre otras y el modelo debe modificarse hasta que se corresponda con las expectativas del usuario. Las adiciones, supresiones o modificaciones de requerimientos deben comprobarse para ver su impacto en el resto del modelo de requerimientos [6].

En el mundo real, sin embargo, la situación es diferente. Los proyectos de software tienen que cumplir presupuestos y plazos. La validación de requerimientos sólo ocupa un pequeño porcentaje del ciclo de vida del proyecto y no puede durar eternamente. Además, los desarrolladores tienden a dejar de lado la especificación de requerimientos para pasar a cuestiones más "difíciles", como el diseño. Por último, los usuarios no siempre tienen todo el tiempo y la voluntad del mundo para colaborar con los analistas en interminables sesiones de validación de requerimientos (inspecciones, recorridos).

Así, el analista se encuentra con limitaciones de tiempo y recursos para llevar a cabo un trabajo crítico como la validación de requerimientos. Como ayudantes de la tarea de validación, el analista puede emplear una serie de herramientas y prácticas, a saber [6]:

- organización lógica del modelo de requerimientos que muestra las interdependencias entre los requerimientos,
- herramientas automatizadas que ayudan a evitar errores administrativos y aceleran algunos aspectos de la validación,
- la comunicación con el usuario en cualquier oportunidad utilizando técnicas como la simulación y la creación de prototipos,
- aportar experiencia en el análisis de sistemas similares; reutilización de modelos de requerimientos anteriores.

Naturalmente, la duración y el esfuerzo dedicados a la validación de requerimientos variarán en función de la aplicación modelada. Además, las herramientas y técnicas de validación pueden parecer más adecuadas para algunos tipos de aplicaciones que para otros. Sin embargo, algún tipo de procedimiento formal de validación es beneficioso para todos los tipos de aplicaciones, ya que "conseguir los requerimientos correctos" es el paso más importante para el éxito de un proyecto de software.

3.2 Técnicas de validación de requerimientos

Hasta la fecha, varios investigadores han propuesto y desarrollado numerosas técnicas de validación de requerimientos para mejorar los procesos de validación de requerimientos y su eficacia, como revisiones, inspecciones, creación de prototipos y métodos formales [36], [38], [39], [40], [41], [42]. Cada una de estas técnicas posee fortalezas y debilidades inherentes sobre otras, por lo que la elección de las técnicas de validación de requerimientos debe hacerse considerando los tipos de sistemas que se están desarrollando y los procesos de desarrollo de software practicados por las organizaciones.

Existen diferentes tipos de técnicas de validación de requerimientos disponibles en la literatura, algunas de ellas son:

Revisión

La revisión de requerimientos es una técnica utilizada por un grupo de personas para validar los requerimientos. Es un proceso formal que involucra lectores del lado del cliente y de los desarrolladores y ayudan a resolver problemas en las primeras etapas del desarrollo del software. El tiempo dedicado a las revisiones de requerimientos se amortiza minimizando los cambios y las alteraciones en el software. Las organizaciones que tienen equipos de revisión independientes generalmente producen sistemas de buena calidad.

Es una de las técnicas más utilizadas para validar los requisitos. El proceso de revisión de requerimientos consta de los siguientes pasos [6]:

- Revisar el plan: se selecciona el equipo, la hora y el lugar para la reunión de revisión.
- Distribuir documentos: los documentos se distribuyen a los miembros del equipo de revisión.
- Preparar la revisión: las personas leen los documentos relevantes antes de la reunión de revisión para encontrar inconsistencias, conflictos, omisiones y otros problemas.
- Reunión de revisión: se discuten los comentarios y problemas individuales, se acuerdan las acciones para abordar el problema.
- Acciones de seguimiento: comprueba si las acciones acordadas se llevan a cabo o no.

- Revisar documento: el documento final se revisa para su aceptación o revisión.

El tiempo requerido para la revisión depende del tamaño del documento. Según [22], se requiere un esfuerzo de 50 personas por horas de un equipo de 4 personas para inspeccionar el documento de requerimientos que tiene 400 requisitos. Deben aclararse los requerimientos que se extravían erróneamente durante la obtención de requerimientos. Si falta información en el documento, es responsabilidad de los ingenieros de requerimientos averiguarlo entre las partes interesadas.

Los requerimientos que no son realistas deben eliminarse o modificarse para el próximo desarrollo del sistema. Los conflictos entre los requerimientos deben negociarse con partes interesadas para resolver los problemas expuestos.

El equipo de revisión debe señalar conflictos, omisiones, inconsistencias y errores en las reuniones de revisión y debe anotar adecuadamente. También pueden verificar los requerimientos para:

- **Verificabilidad**: ¿El requerimiento es comprobable o no?
- **Comprensibilidad**: el usuario final entiende este requerimiento.
- **Trazabilidad**: ¿Es el requerimiento rastreable hasta la fuente de origen?
- **Adaptabilidad**: este requerimiento se puede cambiar sin efectos a gran escala en los otros requerimientos.

Según [43], los siguientes stakeholders deberían participar en el proceso de revisión:

- **Clientes**: son las personas que aportan los requerimientos o las personas que financian el proyecto.
- **Usuarios**: que están involucrados directa o indirectamente con el producto.
- **Analista de requerimientos**: escribe los requerimientos y se comunica con las partes interesadas especialmente con el equipo de desarrollo.
- **Desarrolladores**: participan en el diseño, implementación y mantenimiento del producto.
- Escritores de documentación: participan en la redacción de manuales de usuario y capacitación.
- **Gerente de proyecto**: planifica el proyecto y lidera el equipo de desarrollo para la entrega exitosa del producto.

- **Personal legal**: son responsables del cumplimiento del producto con la regulación y las leyes pertinentes.
- **Programadores**: construyen el producto software.

Inspección

El proceso de inspección es una técnica similar a la revisión que consiste en 6 pasos como la planificación, descripción general, detección de defectos, recolección de defectos, corrección de defectos y seguimiento. Estos pasos se explican en detalle de la siguiente manera [6].

- **Planificación:** el objetivo principal de la fase de planificación es organizar una reunión de inspección donde se determinan los materiales para las inspecciones [44]. Normalmente se seleccionan cuando pasan los criterios de entrada, por ejemplo, cuando la especificación de requerimientos de software está completa [44]. Las actividades claves durante la fase de planificación son, la selección de participantes para la inspección, asignar roles, programar reunión de inspección y distribución de materiales [44].
- **Descripción general:** en esta fase el autor o autores de la especificación de requerimientos de software la explican al equipo de inspección [44]. El objetivo es hacer que la especificación de requerimientos de software sea más fácil de entender para el equipo de inspección. Hay ciertas limitaciones asociadas con la fase general [44]. En primer lugar, estas reuniones consumen esfuerzo y tiempo. En segundo lugar, esta fase no asegura la evaluación independiente de la especificación de requerimientos inspeccionados. Por ejemplo, puede suceder que durante la fase de descripción general todos los inspectores centren su atención en un aspecto particular de la especificación de requerimientos y este conflicto puede consumir más tiempo antes de llegar a una conclusión.
- **Detección de defectos:** la fase de detección de defectos es la fase central del proceso de inspección [44]. En esta fase los defectos son identificados en la especificación de requerimientos de software y se organiza de dos maneras, la primera se realiza la detección de defectos individuales donde cada inspector inspecciona individualmente la especificación de requerimientos y la segunda se

desarrolla en una reunión de grupo donde dos o más inspectores inspeccionan la especificación de requerimientos.

- **Recolección de defectos:** En la fase de recolección de defectos [44], los defectos detectados por los inspectores durante la fase de detección de defectos son recolectados y documentados, se toma la decisión de que el defecto reportado es realmente un defecto o no y se toma la decisión para volver a inspeccionar el artefacto del software defectuoso o no. La idea detrás de los modelos de captura-recaptura en inspecciones de requisitos es permitir que varios inspectores lean la misma especificación de requerimientos [45]. Cada inspector tendrá una lista de requerimientos con defectos. Luego, basado en la superposición de requerimientos defectuosos entre los inspectores, se puede estimar el número de defectos restantes en la especificación de los requerimientos de software.
- **Corrección de defectos:** en la fase de corrección de defectos, el autor o los autores de la especificación de requerimientos de software corrigen la especificación de requerimientos de software en función de los comentarios obtenidos de la fase de recolección de defectos. Esta fase garantiza que los defectos detectados se eliminen por completo de la especificación de requerimientos de software.
- **Seguimiento:** El objetivo de la fase de seguimiento es garantizar que el autor o los autores de la especificación de requerimientos de software haya resuelto todos los requerimientos incompletos o inconsistentes informados [6].

Los roles y responsabilidades en la inspección de requerimientos se expresan de la siguiente manera:

- **Moderador**: es responsable de la selección del equipo y asegura que el equipo realizará sus tareas de acuerdo con cada fase. El moderador es la persona clave en el proceso de inspección.
- **Inspector**: es responsable de encontrar los defectos.
- **Organizador**: planifica todas las actividades dentro o fuera del proyecto.
- **Productor**: es una persona que ha desarrollado artefactos de software, es responsable de explicar los artefactos de software a todos los miembros del equipo.
- **Recorder**: es responsable de anotar todos los defectos en la lista de defectos durante la reunión de inspección. Es un rol opcional.

- **Lector**: se encarga de presentar todo el material y de explicar e interpretar todo el material.

Prototipos

La creación de prototipos es el término utilizado para describir el proceso de construcción y evaluación de modelos de trabajo de un sistema, con el fin de conocer ciertos aspectos del sistema requerido y/o su posible solución. La creación de prototipos es una técnica habitual en las disciplinas de la ingeniería (por ejemplo, la aeronáutica), en la que primero se crea un modelo a escala del artefacto (avión, coche, etc.) que se está produciendo y se utiliza para la experimentación con el fin de llegar a un modelo de producción [6].

En la ingeniería del software, la creación de prototipos se ha defendido como el paradigma para producir software de la forma más rápida y barata posible en alguna fase del desarrollo. Los usos reales del prototipo varían, dependiendo de la fase del ciclo de vida en la que se utilice, así como del modelo de ciclo de vida concreto que se siga. El prototipo de software puede ser desechable (utilizado únicamente para comprender y evaluar soluciones, rendimientos, riesgos, etc.) o puede transformarse en el sistema de producción final [16]. Para que un modelo (de cualquier artefacto, incluido el software) se caracterice como prototipo, debe poder alcanzarse lo siguiente [6]:

- debe ser posible obtener información sobre el comportamiento y el rendimiento del sistema de producción a partir del prototipo. Para proporcionar dicha información, los prototipos deben poder ser totalmente instrumentados; el resultado de dicha instrumentación es que la ejecución de un prototipo da lugar a la generación de datos de los que se puede inferir la información necesaria.
- La creación de prototipos debe ser un proceso rápido. En un entorno bien respaldado, la producción de un prototipo que funcione debería llevar mucho menos tiempo y esfuerzo que la producción de un artefacto a escala real.

La creación de prototipos puede ayudar en las tres fases de la RE, elicitacion, especificación y validación de requerimientos, es decir, al proceso de certificación de la corrección del modelo de requisitos con respecto a la intención del usuario.

Animación

La animación es una técnica que puede utilizarse eficazmente para la validación de sistemas en tiempo real. Los sistemas de tiempo real tienen un conjunto común de nociones como procesos, sincronización de procesos, mensajes y temporizadores. Una preocupación importante en la validación de los sistemas de tiempo real es que los requerimientos de las actividades concurrentes y limitadas en el tiempo se capturen correctamente. Con la llegada de nuevas tecnologías, como las estaciones de trabajo personales de alta velocidad con dispositivos de visualización de alta resolución, se ha hecho posible la representación gráfica de conceptos como los procesos y la visualización dinámica de su comportamiento [6].

En el contexto del desarrollo de sistemas de información, un enfoque de animación de especificaciones de prototipos tiene como objetivo proporcionar un entorno visual para validar y ejecutar simbólicamente las especificaciones conceptuales (en términos de modelos de entidades, procesos y reglas) de un sistema de información. Este enfoque utiliza el término prototipo conceptual para destacar la diferencia entre la ejecución de las especificaciones y la creación de un prototipo del sistema de información. En el prototipo conceptual no es necesario tomar ninguna decisión de diseño antes de tiempo. Al animar el prototipo conceptual, todos los actores del proceso de RE pueden comprender e inspeccionar el comportamiento del sistema en desarrollo lo antes posible [6].

La animación de una especificación es el proceso de proporcionar una indicación del comportamiento *dinámico* del sistema recorriendo un fragmento de la especificación para seguir algún escenario. La animación puede utilizarse para determinar las relaciones causales integradas en la especificación o simplemente como medio de recorrerla para garantizar su adecuación y exactitud mediante el reflejo del comportamiento especificado al usuario [6].

Paráfrasis en lenguaje natural

La técnica de la paráfrasis en lenguaje natural se ha ideado para abordar el problema causado por dos preocupaciones conflictivas en la RE, a saber, la preocupación del analista por desarrollar un modelo de requerimientos formal y la necesidad de los usuarios de comunicar sus requerimientos en su propia terminología. La paráfrasis esuna técnica que permite conciliar las dos preocupaciones proporcionando una versión "fácil de usar" de un

modelo formal de requerimientos. Como la paráfrasis a veces resume partes del modelo de requerimientos, puede ayudar no sólo al usuario, sino también al analista, que puede ver la especificación desde una nueva perspectiva [6].

Al igual que en el caso de la creación de prototipos, no es necesario aplicar la paráfrasis a todo el modelo de requerimientos, ya que puede centrarse en aquellas partes del modelo que necesitan ser aclaradas y reconsideradas por el usuario. Naturalmente, la paráfrasis debe ser una actividad automatizada para que no ocupe más tiempo valioso del necesario. Una herramienta de paráfrasis automática de este tipo ha sido desarrollada por un proyecto de investigación de Myers y Johnson en 1988 y está dirigida a las especificaciones del lenguaje *Gist* [6].

Aspectos importantes a tener en cuenta en la validación de requerimientos:

- La validación es un proceso cuya importancia no puede descartarse fácilmente. Los errores que pasan de la fase de RE al diseño y la codificación pueden tener consecuencias catastróficas en el sistema de software
- La validación de los requerimientos sólo puede realizarse en función de la intención del usuario; por tanto, la participación del usuario en el proceso de validación es primordial
- Nunca se puede demostrar que los requerimientos sean correctos de manera formal. Sin embargo, se pueden comprobar cualidades como la consistencia, la minimización, la no redundancia, entre otros.
- La validación es un proceso que requiere mucho tiempo y que puede ser ayudado en gran medida por herramientas automatizadas.
- Uno de los mayores problemas en la validación de requerimientos es conseguir que el usuario entienda los modelos formales creados por el analista. Una serie de técnicas para superar este problema son la creación de prototipos, la simulación/animación y la paráfrasis de lenguaje natural.
- Todas las técnicas anteriores presuponen el uso de lenguajes y entornos de modelado adecuados en los que los requerimientos pueden acercarse a la experiencia del usuario mediante la ejecución, la animación y la simulación y, por tanto, ser validados más fácilmente
- Se espera que las herramientas expertas desempeñen un papel importante a la hora de validar los requerimientos recurriendo al conocimiento del método y del dominio.

Basado en modelos

La adecuada elección de una técnica de validación basada en modelos es una decisión preliminar de la estructura exploratoria para determinar si un modelo satisface sus requerimientos. La verificación y la validación son esenciales para la gestión de la calidad. Regularmente, las pruebas tienden a confirmar que la verificación cumple con los detalles de la especificación. Al utilizar modelos de casos de uso para pruebas de marco permiten que la descripción satisfaga las necesidades del usuario y mejoran los requerimientos para que sean comprobables y sin ambigüedades. Los casos de prueba generados a partir de los diagramas de actividad ayudan a especificar una base para la prueba. El uso de modelos de casos de uso para pruebas basadas en modelos implica descifrar los modelos de casos de uso con el objetivo final de generar una prueba. Cleaveland sugiere que los errores en el modelo pueden frustrar la capacidad de crear pruebas, antes de crear una prueba, se debe verificar el modelo en busca de errores [46].

En esta técnica se utilizan diferentes modelos para la validación de requerimientos donde la mayoría de las veces los modelos dependen del dominio del proyecto.

Paradigmas de modelado [23].

- Modelado entidad-relación – por ejemplo diagramas de clase (UML-Unified Modeling Language)
- Modelado de flujo de trabajo – por ejemplo, diagramas de actividad (UML), redes de Petri.
- Máquinas de estado - por ejemplo, diagramas de estado, máquinas de estados finitos.
- Lógica de primer orden – por ejemplo, UML-OCL (Object Constraint Language), VDM (Vienna Development Method). Se puede usar como complemento con los otros paradigmas anteriores, proporcionando información sobre objetos de datos y relaciones. O puede usar solo, expresando modelos estructurales y modelos de comportamiento.

Basado en puntos de vista

Según Sommerville [47], el enfoque basado en puntos de vista para la RE identifica que una sola perspectiva de los requerimientos del sistema no puede descubrir toda su

información relacionada. Debe haber un número diferente de puntos de vista, que se consideran para recopilar y organizar todos los requerimientos.

El objetivo de la validación de requerimientos basado en punto de vista es identificar y clasificar los problemas relacionados con la corrección, integridad e inconsistencia [9]. Una encapsulación de información parcial sobre los requerimientos de un sistema se puede llamar como un punto de vista [47]. El proceso de resolución de puntos de vista se ocupa de identificar las diferencias entre dos puntos de vista diferentes, clasificar y evaluar esas discrepancias e integrar diferentes soluciones en una sola representación [48].

De acuerdo con Leite [48], este enfoque contiene ciertos conceptos que es necesario comprender para la validación de requerimientos basado en punto de vista. Estos conceptos son:

a) El Universo del discurso es el contexto general para desarrollar el software. Junto con esto, contiene todas las fuentes de información, incluidas todas las personas asociadas al software.
b) Las personas son conocidas como actores en el universo del discurso.
c) Un punto de vista se conoce como una posición mental de un individuo mientras examina u observa el universo del discurso.
d) Una perspectiva es un conjunto de hechos observados y modelados de acuerdo con un aspecto de modelado particular y un punto de vista.
e) La relación entre entidades y atributos se conoce como hecho.
f) Una vista es una integración de perspectivas y jerarquías.
g) Hay dos tipos de conceptos de jerarquía que se utilizan, conocidos como jerarquías "es-a" y "parte de" en el universo del discurso.

Se utilizan tres perspectivas (perspectivas de datos, actores y procesos) y dos jerarquías (es-a, parte de) para construir una vista, donde un analista de sistemas explica el problema. Después de la comparación de perspectivas y jerarquías, se crea una lista de discrepancias y tipos de discrepancias. Cuando hay una cadena de análisis disponible, cada analista resuelve los conflictos internos e integra su percepción final en una vista, que describe la perspectiva y las jerarquías juntas [48]. Las vistas se toman desde los puntos de vista, que son necesarios para identificar y clasificar las discrepancias entre los diferentes puntos de vista para su corrección e integridad.

Casos de prueba

El objetivo de las pruebas de requerimientos es validar los requerimientos de la especificación de requerimientos de software en lugar del sistema. Los casos de prueba se definen para cada requerimiento de la especificación de requerimiento de software, esto ayuda a identificar requerimientos incompletos o ambiguos porque si para un requerimiento en particular es difícil de derivar un caso de prueba, entonces indica algún problema [39].

Según Sommerville [23], el requerimiento debe estar escrito de tal manera que el diseño de casos de prueba para cada uno de los requerimientos sea fácil. Entonces, como resultado, los desarrolladores pueden verificar que el requerimiento sea implementable o no. El propósito de sugerir casos de prueba para requerimiento es validar los requerimientos en lugar de validar el sistema. Al definir los casos de prueba para los requerimientos, se deben tener en cuenta algunas consideraciones importantes [9]:

a) El escenario de uso de los requerimientos define el contexto en el que se debe aplicar la prueba.
b) Es necesario asegurarse de que el requerimiento, por sí solo, contenga suficiente información para definir el caso de prueba.
    - Si la información no está disponible en ese requerimiento específico, entonces se deben considerar otros requerimientos relacionados para obtener esta información.
    - En caso de obtener información de otros requerimientos, debe registrarse el requerimiento para el que se busca la información.
    - Estos requerimientos pueden depender unos de otros, por lo que es necesario registrarlos.
c) Siempre verifique que el requerimiento esté utilizando casos de prueba únicos o múltiples. Junto con esto, también es posible que haya más de un requerimiento descrito en una sola descripción de requerimientos.
d) Los casos de prueba deben ser obvios si se vuelve a establecer el requerimiento.

Los requerimientos deben ser comprobables. Si las pruebas para los requerimientos se diseñan como parte del proceso de validación, esto revela con frecuencia problemas en los requerimientos. Si una prueba es difícil o imposible de diseñar, esto generalmente significa que los requerimientos serán difíciles de implementar, por lo que deberían reconsiderarse.

El desarrollo de pruebas a partir de los requerimientos del usuario antes de escribir cualquier código es una pieza integral de la programación extrema [23].

Kotonya [22] sugiere que debe diseñarse y completarse un formulario de registro de prueba para todos y cada uno de los requerimientos que se prueba con la siguiente información:

- El identificador del requerimiento.
- Requerimientos relacionados.
- Descripción de la prueba.
- Problemas de requerimientos.
- Comentarios y recomendaciones.

3.3 Roles en la validación de requerimientos

Las personas involucradas en la validación de requerimientos son denominados actores del proceso de RE, no se identifican individualmente sino por su función.

La RE establece los siguientes actores interesados en el problema o su solución:

- Clientes, usuarios, expertos en dominios.
- Ingenieros de software, ingenieros de requerimientos.
- Gerentes de proyectos.

Kotonya y Sommerville [23] indican que el papel de las personas durante el proceso de RE está representado en el modelo. Este último relaciona el proceso de RE con los stakeholders que comienza con la actividad "Comprender el problema del negocio" involucra a ingenieros de requerimientos, expertos en dominio y usuarios finales, la actividad "Establecer esquema de requerimientos" es cumplida por ingenieros de requerimientos, usuarios finales, seguido de la actividad "Seleccionar un sistema de prototipado" realizada por ingenieros de software, gerentes de proyecto, "Ejecutar el prototipo de desarrollo de actividades" por ingenieros, ingenieros de software, por último evaluar la actividad "Prototipo completo" por el usuario final, expertos en el dominio, ingenieros de requerimientos, ingenieros de software.

Sourour [58] en el desarrollo de la metodología para la validación de requerimientos colaborativos en un entorno cooperativo establece en detalle los roles y habilidades de los

stakeholders, los resultados de cada actividad en el proceso de validación y los medios para su implementación. Además, sugiere seleccionar algunos participantes fuera del equipo de desarrollo, debido a que son más objetivos y se involucran menos en el proyecto. En consecuencia, es necesario realizar la asignación de roles, es probable que un stakeholder pueda asumir varios roles. Los roles y competencias se resumen en la Tabla 2.

Tabla 2. Roles y competencias de los stakeholders en la validación de requerimientos.

Participante	***Intervención***	***Roles***	***Competencias y experiencia***
Analista	Proceso completo	Preparar las reuniones y asegurar la secuencia de pasos. Asegurar la realización de los objetivos comerciales y mantener las relaciones para no descuidar los recursos humanos.	Análisis de sistemas de información, manejo de recursos humanos, comunicación, orden, decisión y negociación.
Cliente	Validación	Identificar necesidades, leer los requerimientos para verificar la correspondencia con las necesidades.	Comunicación.
Gerente de Proyectos	Inspección	Utilizar las especificaciones para planificar el suministro y el proceso de desarrollo del sistema.	Gestión de costo del problema de dominio, demora, comunicación y técnica.
Experto en comunicación y soluciones de dominio.	Validación	Identificar los requerimientos funcionales	Comunicación de problemas y soluciones de dominio

Participante	Intervención	Roles	Competencias y experiencia
Usuario final	Validación	Difundir requerimientos funcionales y no funcionales, organización, contexto, restricción	Dominio de problemas y conocimiento de soluciones.
Ingeniero y desarrollador de sistemas	Verificación	Utilizar los requerimientos para comprender el sistema en desarrollo.	Comunicación HMI (Human Machine Interface)
Ingeniero de prueba de sistemas	Verificación	Utilizar los requerimientos para desarrollar pruebas de validación del sistema.	Habilitad en la comunicación de las pruebas.
Ingeniero de mantenimiento de sistemas	Mantener la validación	Utilizar los requerimientos para ayudar a comprender el sistema.	Comunicación
Diseñador	Verificación	Utilizar los requerimientos para detallar y completar las versiones.	Comunicación, dominio de la solución

Capítulo 4 Selección de enfoques

Este capítulo describe una revisión bibliográfica de la literatura en una revisión bibliográfica de las principales tendencias de la Validación de Requerimientos del software desde el año 2007 hasta el año 2021 [4]. La revisión está centrada en la etapa de validación de requerimientos en el campo de la ingeniería de software para investigar los enfoques propuestos por los estudios previos. En la Figura 5 se muestra el alcance de la revisión bibliográfica.

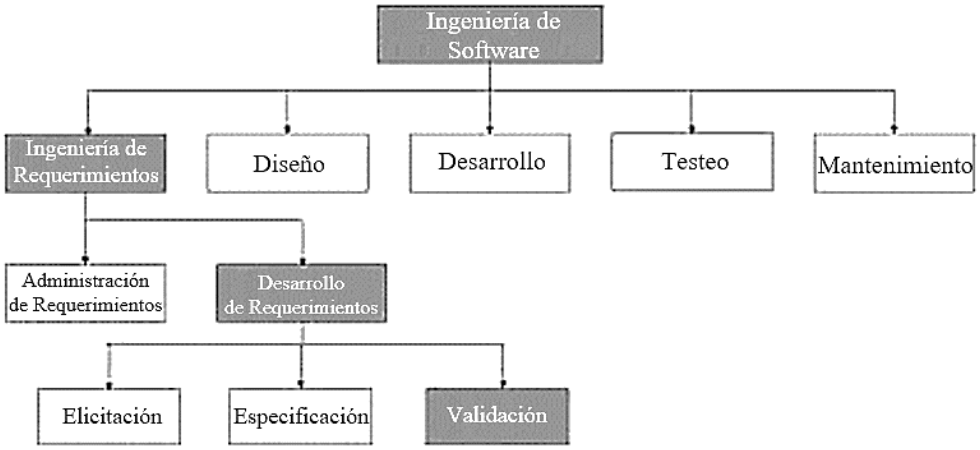

Fig. 5. Mapa de la revisión bibliográfica.

Se adaptó el protocolo de estudio de mapeo sistemático realizado por [49] que incluye preguntas de investigación, decisiones para la estrategia de búsqueda, estrategia de extracción de datos, criterios de inclusión/exclusión y estrategia de síntesis de datos. A continuación, se describen las actividades del proceso de revisión bibliográfica.

4.1 Protocolo de búsqueda de artículos

Para dar comienzo a la revisión bibliográfica se planteó la siguiente pregunta ¿Cuáles son las principales prácticas relacionadas con el proceso de validación de requerimientos? Esta pregunta de investigación tuvo como objetivo identificar las principales prácticas sobre el proceso de validación de requerimientos.

Para las búsquedas se desarrollaron frases, en inglés y español, con el fin de encontrar respuestas a la pregunta inicial, que son las siguientes:

- (“requirements”) y (“validation” o “validate” o “validity”) y (“methodology” o “technique” o “method” o “tool”)
- ("requisitos" o “requerimientos”) y ("validación" o "validar" o "validez") y ("metodología" o "técnica" o "método" o "herramienta")

4.2 Búsqueda de artículos

Se empleó un proceso de búsqueda automatizado para encontrar todos los estudios relevantes de validación de requerimientos. Se realizaron las búsquedas en cuatro fuentes de datos principales:

- The Institute of Electrical and Electronics Engineers (IEEE)
- ScienceDirect
- SpringerLink
- Association for Computing Machinery (ACM Digital Library)

Los artículos buscados incluyen revisiones por pares:

- Journal papers.
- Conference papers.
- Workshop papers.

Las búsquedas se aplicaron sobre el título y resúmenes de los artículos publicados en español e inglés y en las bases de datos desde el año 2007.

4.3 Selección de artículos

Para el proceso de selección se establecieron varios criterios de inclusión y exclusión de artículos:

- Los artículos seleccionados están directamente relacionados con el tema de validación de requerimientos en el área de RE.

- Los artículos que discuten sobre la validación de requerimientos en la fase de prueba del software desarrollado o en la implementación en prueba con respecto a los requerimientos, no fueron considerados.
- De los artículos que utilizan términos como "Revisión sistemática de la literatura", "Revisión de la literatura", "Análisis sistemático", sólo fueron considerados aquellos que proponen nuevos enfoques de validación de requerimientos.

4.4 Evaluación de artículos.

Para la evaluación de los artículos se elaboró una lista de criterios que fue adaptada de [6], la misma permitió seleccionar los artículos finales en base a cuatro criterios que se enumeran en la Tabla 3. Las preguntas incluidas en la lista de criterios se responden con "SI", "PARCIAL" o "NO", calificado como 2, 1 o 0, respectivamente. La suma de las puntuaciones de todas las preguntas se utilizó para evaluar la calidad de un artículo.

Tabla 3. Criterios de evaluación.

Sección	Criterios
Introducción	• ¿La introducción proporciona una descripción general del aporte para la validación de requerimientos? • ¿Está claramente definido el propósito / objetivo de la investigación?
Metodología	• ¿Está claramente definida la metodología de investigación? • ¿Se establece una sucesión clara de pruebas a partir de las observaciones a las conclusiones? • ¿Se indican claramente los procedimientos de análisis?
Resultados	• ¿Se indica adecuadamente el caso y su contexto? • ¿El estudio de caso se basa en la teoría y está vinculado a la literatura? • ¿Están los hallazgos claramente establecidos? ¿Los resultados ayudan a resolver los problemas de validación de requerimientos?

Sección	Criterios
	• ¿Los resultados son adecuados para la audiencia, fácil de leer y bien estructurado?
Conclusión	• ¿Contiene conclusiones claras, implicaciones para la práctica y la investigación futura? • ¿Las conclusiones, implicaciones para la práctica y futuras investigaciones son derivadas adecuadamente para su audiencia? • ¿Existen límites o restricciones impuestas a la afirmación de conclusión?

El proceso general del estudio de mapeo se ha dividido en cuatro fases: la fase 1 involucra la pregunta de investigación y la formalización de la estrategia de búsqueda, la fase 2 involucra la búsqueda de las referencias de las bases de datos usando las cadenas de búsqueda y luego eliminar duplicados, la fase 3 implica la selección de artículos y la extracción de datos, mientras que la fase 4 implica la evaluación de la calidad de los artículos. El proceso del estudio de mapeo se muestra en la Figura 6.

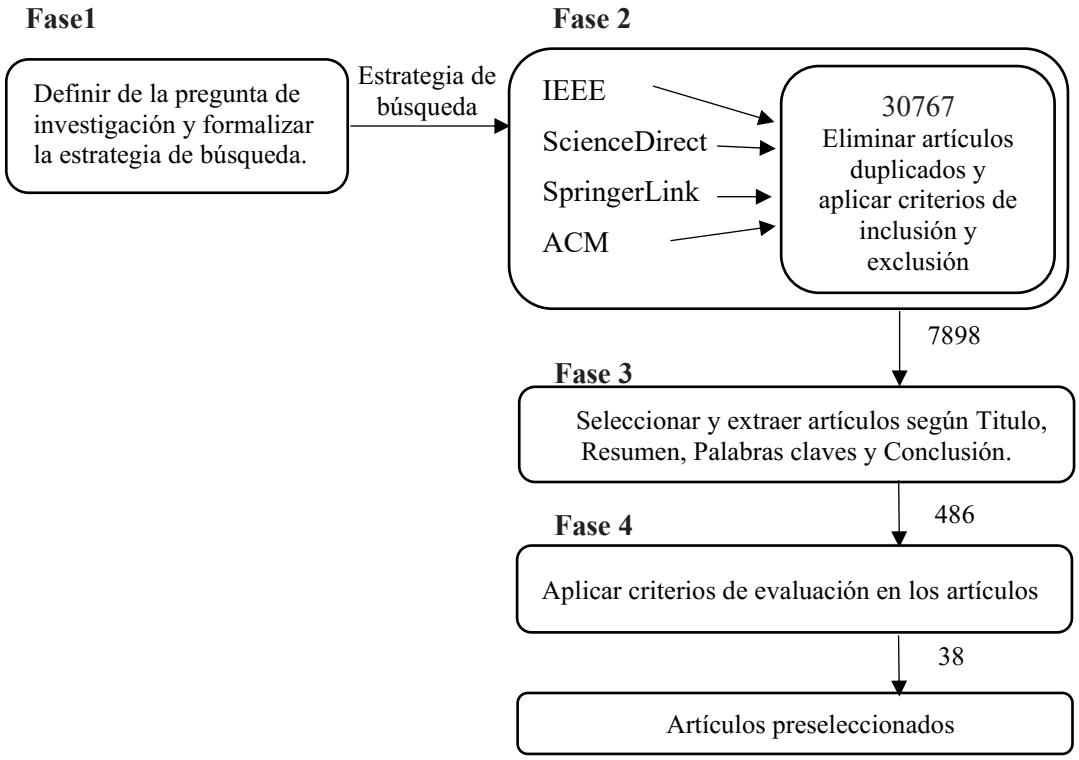

Fig. 6. Proceso del estudio de mapeo

En la búsqueda inicial aplicando la estrategia de búsqueda se encontraron 30767 artículos, luego de eliminar los artículos duplicados y aplicar los criterios de inclusión y exclusión definidos en la sección 4.4 se preseleccionaron 7898 artículos. Después de leer los títulos, los resúmenes, palabras claves y las conclusiones se preseleccionaron 486 artículos relevantes. Por último, luego de aplicar los criterios de evaluación definidos en la Tabla 3 se preseleccionaron 38 documentos.

Para cada uno de los artículos, se identificó el año de publicación, el nombre y el dominio de la aplicación. En la Tabla 4 se presentan los resultados.

Tabla 4. Artículos de la revisión bibliográfica.

Ref. Bibliog.	***Año***	***Nombre***	***Dominio de Aplicación***
[50]	2007	FBCM	Empresa
[51]	2007	CPN Tool	Cuidado de la salud
[52]	2008	AutoPA3.0	Sistema de librería
[53]	2009	EuRailCheck	Transporte (tren)
[54]	2009	ACE Framework	Discusión en línea /foro
[55]	2009	-	Seguridad crítica
[56]	2010	SQ^(2)E	Línea de producción
[57]	2010	-	Documento de dominio genérico
[58]	2011	CoReVDO	Sistemas distribuídos
[59]	2011	MaramaAI	Sistema ATM
[60]	2011	-	Sistema electrónico de vehículos
[61]	2011	VRP	Software embebido
[62]	2012	-	Sistema de biblioteca
[63]	2012	Othello	Transporte

Ref. Bibliog.	*Año*	*Nombre*	*Dominio de Aplicación*
[65]	2013	-	Sistema de multiagente
[66]	2014	CuRV	Smart Phone
[67]	2014	SpecQua	Sistema cuidado de la salud
[68]	2014	-	Empresa
[69]	2015	ReVAMP	Empresa
[70]	2015	-	Empresa Shopping On line
[71]	2015	SimTree	Dispositivo sanitario
[72]	2016	-	Transporte
[73]	2016	MobiMEReq	Aplicaciones móviles
[74]	2016	ASATF	Sistema de búsqueda.
[75]	2016	TestMEReq	Empresa
[76]	2017	-	Señalización ferroviaria
[77]	2017	SHPbench	Asistencia al conductor de automóviles
[78]	2018	-	Sistema de control para nanosatélite y satélite
[79]	2018	-	Sistema de control de proceso.
[80]	2018	-	Sistema de asistencia al conductor
[81]	2019	-	Institución educativa
[82]	2019	RM2PT	Empresa
[83]	2019	-	Sistema de gestión de curso
[84]	2020	ValFAR	Sistema de gestión de cursos
[85]	2020	-	Industria automotriz

Ref. Bibliog.	*Año*	*Nombre*	*Dominio de Aplicación*
[86]	2020	-	Sistemas de aviación civil
[87]	2021	-	Sistema de control de trenes

Capítulo 5 Estudio realizado

Este capítulo describe un análisis de enfoques, siguiendo la misma línea de investigación realizada en [4] y [5], que consiste en obtener información de los enfoques sobre las características asociadas al proceso de validación de requerimientos en el ciclo de vida del software: la naturaleza de la información. En primer lugar, se responden las preguntas Quién, Qué, Cuando, Por qué y Cómo se validan los requerimientos en los enfoques seleccionados y luego se realiza un análisis comparativo entre las distintas características de los enfoques.

5.1 Análisis de enfoques

Según Santana [4], evalúa cuatro metodologías de la revisión con el marco "Way-of" o "forma de" [28] debido que permite modelar los procesos de negocio y el modelo de referencia para revisiones técnicas [29] donde se identificaron las siguientes contribuciones al proceso de Validación de Requerimientos:

- Funciones, componentes, entornos y características.
- Actividades de planificación.
- Técnicas de control e indicadores de rendimiento.
- Definición de estándares.
- Aceptación del cliente/usuario.
- Dominios de aplicación.
- Participación cliente/usuario.
- Etapas del ciclo de vida del desarrollo del software donde validan los requerimientos.

Del análisis de los resultados obtenidos en la evaluación de las metodologías para la validación de requerimientos [4], en [5] se extraen diferentes características que son calificadas como puntos de evaluación para los enfoques seleccionados. Estas características se resumen en términos de recomendaciones de buenas prácticas que pueden mejorar el conocimiento en el desarrollo de enfoques para validación de requerimientos.

Recomendaciones de buenas prácticas

Definición: Definir la función, los componentes, el entorno, los componentes del entorno y las características del enfoque.
Modelado: Especificar el modelo, los componentes del modelo y las relaciones entre los componentes del modelo del enfoque.
Operación: Definir la planificación, fases, actividades, tareas, requerimientos que valida, roles de las tareas y funciones y responsabilidades de los roles del enfoque.
Control: Definir la forma de control, indicadores de rendimiento y objetivos medidos por indicadores del enfoque.
Gestión de defectos: Definir la forma de identificar, documentar y seguir los defectos del enfoque.
Aceptación: Definir la aceptación del cliente sobre las especificaciones, la aceptación del cliente sobre las condiciones, la aceptación del cliente sobre las restricciones, la aceptación del cliente sobre los parámetros de calidad y la validación de los requerimientos en el ciclo de vida del desarrollo de software del enfoque.
Soporte: Definir técnicas, herramientas y estándares del enfoque.
Dominio de aplicación: Definir el dominio de aplicación del enfoque.

Continuando con la misma línea de investigación este trabajo obtiene información sobre las características asociadas al proceso de validación de requerimientos en el ciclo de vida del software: la naturaleza de la información Quién, Qué, Cuando, Por qué y Cómo validar los requerimientos. La información se obtiene a partir de un análisis de los enfoques para estudiar sus características, necesidades de información y restricciones en base a la lista de criterios de la Tabla 5.

Tabla 5. Lista de criterios.

Criterios	***Descripción***
Quién	Las partes interesadas en la validación de los requerimientos de software y su participación en el proceso.
Qué	Los diversos tipos de documentos, modelos o planificaciones que deben definirse para la validación.

Criterios	*Descripción*
Cuando	La validación de los requerimientos a lo largo del ciclo de vida del desarrollo de software.
Por qué	Los beneficios de realizar la validación de los requerimientos.
Cómo	Las técnicas e indicadores para validar los requerimientos.

En el Anexo 1 se presenta el desarrollo del análisis de cada uno de los enfoques seleccionados de la Tabla 4. El análisis consiste en responder a cada una de las preguntas de la Tabla 5 para cada enfoque.

5.2 Resultados obtenidos

A partir de la Tabla del Anexo 1 se realiza un análisis comparativo entre los distintos enfoques para la validación de requerimientos, donde se extraen los siguientes resultados:

En la Tabla 6 se muestra los resultados según los roles de los stakeholders definidos por los enfoques analizados.

Tabla 6. Roles de los stakeholders de los enfoques.

Roles de los stakeholders	***Total Enfoques***	***% sobre el total***
analista	4	**19%**
cliente	3	**14%**
gerente de proyectos	0	**0%**
experto en dominio	1	**5%**
usuario	4	**19%**
ingeniero - desarrollador	9	**43%**
Ingeniero de pruebas	0	**0%**
Ingeniero de mantenimiento	0	**0%**
diseñador	0	**0%**

Según el análisis realizado el rol de ingeniero-desarrollador es el más definido por los enfoques.

La literatura menciona varios enfoques para la validación de los requerimientos, pueden clasificarse de acuerdo con el nivel de formalidad de la especificación con lo que comienza el proceso de validación de la especificación formal, semiformal e informal. La especificación más utilizada es la semiformal, utiliza modelos UML, diagrama estructurado, descripción textual/escenario, lógica de descripción y otros componentes. La mayoría de las especificaciones semiformales implica el uso de modelos / diagramas como modelo UML. Un modelo UML se describe como la representación de diseño del código fuente y este diagrama es útil para hacer comprensible el código fuente [72]. Las especificaciones semiformales también reciben gran interés porque los modelos se descomponen fácilmente en partes más pequeñas y esto permite que se entiendan mejor [73]. Otra especificación utilizada para representar los requerimientos es la especificación informal escrita en lenguaje natural. La representación menos utilizada es la especificación formal. Esto se debe a que el uso de la especificación formal es desafiante y difícil debido a que tiene una base rigurosa en matemática [75].

La especificación puede tener una solución automática, semiautomática o manual y generalizada en tres niveles de evaluación: un nivel interno, esta actividad es proporcionada por el equipo de RE, un nivel externo es atractiva para los clientes, y un nivel mixto que involucra el equipo de trabajo acompañado por el cliente [49]. En la Tabla 7 se muestran los resultados según los niveles de formalidad, solución y evaluación de los enfoques analizados.

Tabla 7. Nivel de formalidad, solución y evaluación de los enfoques.

	Total enfoques	***% sobre el total***
Nivel de formalidad		
Formal	3	**8%**
Semiformal	32	**84%**
Informal	3	**8%**

	Total enfoques	*% sobre el total*
Nivel de solución		
automática	5	**13%**
semiautomática	33	**87%**
Nivel evaluación		
interna	31	**82%**
mixta	7	**18%**

Según el análisis realizado las especificaciones semiformales con soluciones semiautomáticas y evaluaciones mixtas es ampliamente utilizada por los enfoques.

La Tabla 8 muestra los resultados según las etapas del proceso de desarrollo de software cuando validan los requerimientos los enfoques analizados.

Tabla 8. Etapas de validación de requerimientos de los enfoques.

Etapas del proceso de desarrollo del sistema	***Total Enfoques***	***% sobre el total***
elicitacion	15	**30%**
análisis	8	**15%**
especificación	22	**42%**
modelado	2	**4%**
diseño	3	**5%**
mantenimiento	0	**0%**
ciclo de vida del desarrollo del sistema	2	**4%**

Según el análisis realizado la mayoría de los enfoques validan los requerimientos en la especificación de requerimientos y en la elicitacion.

Existen diversos criterios de calidad con el propósito de lograr un buen documento de requerimientos, los mismos fueron mencionaron en la sección 3.1. En la Tabla 9 se

muestran los resultados según los criterios de validación de requerimientos definidos por los enfoques analizados.

Tabla 9. Criterios de calidad de los enfoques.

Criterios de calidad	***Total Enfoques***	***% sobre el total***
corrección	20	**30%**
completitud	12	**17%**
consistencia	20	**30%**
ambigüedad	5	**8%**
verificabilidad	1	**2%**
trazabilidad	5	**8%**
comprensibilidad	1	**2%**
legibilidad	0	**0%**
prioridad	0	**0%**
validez	2	**3%**
modificabilidad	0	**0%**

Según el análisis realizado la mayoría de los enfoques consideran como criterio de calidad en la especificación de requerimientos la corrección, la consistencia y la completitud.

En relación con la clasificación de técnicas para la validación de requerimientos presentada en la Sección 3.2, en la Tabla 10 se muestran los resultados según las técnicas utilizadas por los enfoques analizados.

Tabla 10. Técnicas de validación de los enfoques.

Técnicas para validación	***Total Enfoques***	***% sobre el total***
revisión	6	**12%**
inspección	7	**14%**
prototipos	10	**19%**
animación	4	**9%**
paráfrasis en lenguaje natural	7	**14%**

Técnicas para validación	*Total Enfoques*	*% sobre el total*
basado en modelos	9	**17%**
basado en puntos de vista	0	**0%**
casos de prueba	8	**15%**

Según el análisis realizado la mayoría de los enfoques proponen técnicas de validación de requerimientos como prototipos, basado en modelos y casos de prueba. Donde, muchos de los enfoques utilizan combinación de técnicas.

La Tabla 11 muestra una síntesis de los resultados obtenidos en el análisis realizado a los enfoques especificando Quién, Qué, Cuando, Por qué y Cómo validar los requerimientos.

Tabla 11. Síntesis del análisis a los enfoques de validación de requerimientos.

	% enfoques	**Descripción**
Quién	**43%**	Ingeniero- desarrollador.
Qué	**84%**	Especificación de requerimientos semiformal con solución semiautomática y evaluación interna.
Cuando	**36%**	Etapa de especificación y elicitación de requerimientos.
Porqué	**26%**	Corrección, consistencia y completitud.
Cómo	**19%**	Técnicas prototipo y basadas en modelos. Combinación de técnicas.

Capítulo 6 Conclusiones y Trabajo Futuro

La literatura tiende a considerar la validación de requerimientos como un proceso heterogéneo basado en la aplicación de una variedad de técnicas independientes; sin poder especificar: Quién, Qué, Cuando, Por qué y Cómo validar los requerimientos.

Este trabajo se ha focalizado en un proceso fundamental de la Ingeniería de Requerimientos: la Validación. Se han comparado 38 trabajos como contribuciones al proceso de Validación de Requerimientos para obtener información de los enfoques y analizar sus características, necesidades de información y restricciones.

Si bien los enfoques cumplen con la mayoría de las características de la naturaleza de la información Quién, Qué, Cuando, Por qué y Cómo validar los requerimientos, se encontraron algunas características que deben considerarse al momento de aplicarlos.

Todos los enfoques fueron desarrollados para dominios específicos de aplicación.

Los enfoques proporcionan diferencias en la visión de dominio para validar los requerimientos con diversos grados de éxito. Este éxito depende de la naturaleza de la organización en sí y del conocimiento profundo del negocio para adaptar el enfoque a las necesidades del negocio y del usuario.

Se observa la escasa participación de los usuarios/clientes en el proceso de validación de requerimientos. La mayoría de los enfoques realizados fueron principalmente para la comprensión y la responsabilidad de los ingenieros de requerimientos y muy limitado el número de enfoques que proporcione la confirmación de la consistencia y la validación de los requerimientos del lado de los clientes.

Los enfoques realizan la validación de requerimientos sobre la especificación de requerimientos, es decir, no se aplican en las diferentes etapas del ciclo de vida del desarrollo del software. Si bien un enfoque emplea la validación de los requerimientos en todo el ciclo de vida del sistema esto se debe al uso de estándar de aviación ARP4754A [90], donde respalda la certificación de sistemas de aeronaves, abordando "el ciclo completo de desarrollo de aeronaves, desde los requerimientos del sistema hasta la verificación de los sistemas". En el marco de la observación anterior, el cumplimiento de

estándares es insuficiente en el tratamiento de las características implícitas esperadas en el desarrollo de software profesional.

Los enfoques realizan la validación de requerimientos a especificaciones semiformales implementando una solución semiautomática con evaluación interna centrada en el rol de ingeniero-desarrollador. Las especificaciones semiformales reciben gran interés debido a que los modelos pueden descomponerse fácilmente en partes más pequeñas y esto permite que se comprendan mejor [91]. En cuanto a los roles, los participantes son seleccionados en base a la participación de la validación en varias fases [92].

La mayoría de los enfoques realizan la validación de requerimientos en la especificación de requerimientos considerando múltiples criterios de calidad de los requerimientos, los criterios más importantes son la corrección, consistencia y completitud. Además, la mayor parte de los enfoques no verifican la consistencia de los requerimientos en todo el ciclo de vida del desarrollo del software, sino que tienden a emplear una verificación de consistencia parcial que solo se enfoca desde la elicitación de requerimientos hasta el modelado/especificación de requerimientos.

Si bien la técnica más utilizada por los enfoques es el prototipo, la mayoría de los enfoques utilizan una combinación de técnicas, esto se debe a que cada técnica de validación de requerimientos tiene sus objetivos y la participación de diferentes partes interesadas [92].

La mayoría de los enfoques analizados fueron realizados principalmente para la comprensión y la responsabilidad de los ingenieros y pocos enfoques proporcionan la confirmación de la consistencia y la validación de los requerimientos del lado del cliente.

A partir de los resultados obtenidos de [4] y [5], y de los resultados del presente trabajo, donde se analizaron enfoques en base a la naturaleza de la información, se mejoró el conocimiento en el desarrollo de enfoques para validación de requerimientos. Por todo lo mencionado anteriormente se propone como trabajo futuro desarrollar un enfoque que puede ayudar en el proceso de validación de requerimientos. El enfoque a desarrollar, en primer lugar, debe describir un marco de referencia para el desarrollo de la estructura y en segundo lugar establecer un modelo para el proceso de validación de requerimientos a nivel de caja blanca en la estructura interna del diseño de los componentes del desarrollo del

sistema. Esto permite garantizar la calidad del producto o sistema, enfocándose en las diferentes etapas del ciclo de vida del software.

Bibliografía

1. F. Brooks: The Mythical Man Month: Essays on Software Engineering. Reading, Mass, USA, Addison-Wesley, (1975).
2. IEEE: SWEBOK Guide V3.0, Piscataway: IEEE, (2014).
3. E. Gottesdiener: Software Requirements Memory Jogger: A Pocket Guide to Help Software and Business Teams Develop and Manage Requirements. Methuen, MA: Goal/QPC, (2005).
4. S.R. Santana, L. Antonelli, P. Thomas: Evaluación de metodologías para la validación de requerimientos, XXVII Congreso Argentino de Ciencias de la Computación (CACIC), pp. 419-428, ISBN 978 -987-633-574-4, (2021).
5. S.R. Santana, L.R. Antonelli, P.J. Thomas: Best Practices for Requirements Validation Process. In: Pesado, P., Gil, G. (eds) Computer Science – CACIC 2021, CACIC 2021, Communications in Computer and Information Science, vol 1584, Springer, Cham, https://doi.org/10.1007/978-3-031-05903-2_10, (2022).
6. P. Loucopoulos, V. Karakostas: System Requirements Engineering, McGraw-Hill, London, ISBN 0-07-707843-8 (1995).
7. W. W Royce: Managing the Development of large software systems: Concepts and Techniques In Proc. WESCON, August (1970).
8. D. McCracken, M. Jackson: Life Cycle concept considered harmful. ACM SIGSOFT Soft. Eng. Notes, vol. 7, No. 2, April (1982).
9. M. Alavi.: An Assessment of the Prototyping Approach to Information Systems Development. Communications of the ACM, (1984).
10. M. Jackson.: Systems Development. Prentice Hall. Martin, (1982).
11. S. Castano, V. De Antonellis: Reuse of Conceptual Requirement Specifications, IEEE International Symposium on Requirements Engineering, IEEE Computer Society Press, San Diego, California, pp. 121-124, (1993).
12. Balzer: RADC System/Software Requirements Engineering Testbed Research and Development Program. Report TR-88-75, Rome Air Development Center, Griffiss Air Force Base, N.Y., June, (1988).
13. R. Yeh, P. Ng: Software Requirements-A management perspective. In System and Software Requirements Engineering. R. H. Thayer & M. Dorfman (eds) IEEE Computer Society Press, (1990).
14. B.W. Boehm: A Spiral Model of Software Development and Enhancement, IEEE COMPUTER Vol. 21 No 5, May, (1988).
15. C. Floyd.: A Systematic look at prototyping. In Approaches to Prototyping. Budde, R. ed., Springer-Verlag, (1984).
16. P. Zave: The Operational Versus the Conventional Approach to Software Development, Communications of the ACM, Vol. 27, No. 2, February (1984).
17. R. Balzer, T.E. Cheatham, C. Green: Software Technology in the 1990's: Using a New Paradigm, IEEE COMPUTER, November (1983).
18. P.A. V. Hall: Domain Analysis. Proc. UNICOM Seminars, London, December (1991).
19. P. A. Laplante: Requirements Engineering for Software and Systems, CRC Press (2019).
20. B. H. C. Cheng, J. M. Atlee: Current and Future Research Directions in Requirements Engineering, Design Requirements Engineering A Ten-Year Perspective, Lecture Notes in Business Information Processing, vol. 14, pp. 11–43 (2019).

21. S. L. Pfleeger: Software Engineering – Theory and Practice, Prentice Hall (1998).
22. S. R Pressman: Ingeniería del software. México D.F.: Mc Graw Hill, (2010).
23. G. Kotonya, I. Sommerville: Requirements Engineering: Processes and Techniques, JohnWiley & Sons, England, (1998).
24. G. Kotonya, I. Sommerville: Requirements Engineering: Processes and Techniques, John Wiley & Sons, (2004).
25. A. Terry Bahill, Steven J. Henderson: Requirements development, verification, and validation exhibited in famous failures, Systems Engineering. 8. 1 - 14. 10.1002/sys.20017 (2005).
26. J. N. Martin: Overview of the EIA 632 standard: processes for engineering a system, 17th DASC. AIAA/IEEE/SAE. Digital Avionics Systems Conference. Proceedings (Cat. No.98CH36267), pp. B32-1, (1998).
27. S.R. Pressman.: Ingeniería del software. Un enfoque práctico, Séptima edición, México D.F., Mc Graw Hill, (2010).
28. B. Boehm, V. Basili: Software defect reduction top 10 list, IEEE Computer, (2001).
29. F. Shull: What we have learned about fighting defects, in Proceedings of the International Software Metrics Symposium, pp. 249-258, (2002)
30. K. Wiegers: Creating a Software Engineering Culture, Dorset House Publishing Company, Aug. (1996).
31. K. Wiegers: Software Requirements, 2nd ed. Microsoft Press, (2003).
32. P. Skokovi, M. R. Skokovi: Requirements Based Testing Process Overview (Originally presented as “Getting it right the first time”), International Journal of Industrial Engineering and Managem ent (IJIEM), vol. 1, n.o 4, pp. 155 - 161, (2010).
33. Bender RBT Inc: Requirements Based Testing Process Overview. Bender RBT Inc, (2009).
34. H. Hofmann, F. Lehner: Requirements engineering as a success factor in software projects, IEEE Software, vol. 18, no. 4, pp. 58-66, (2001).
35. L. Kof, R. Gacitua, M. Rouncefield, P. Sawyer: Ontology and Model Alignment as a Means for Requirements Validation. IEEE Fourth International Conference on Semantic Computing, 2010, pp. 46-51, (2010).
36. T. Pyzdek.: Quality engineering handbook. New York: Marcel Dekker, (2000).
37. Dubois, E. & Hagelstein, J.: Reasoning on Formal Requirements: A Lift Control System. Proc. 4th Int'l Workshop on Software Specification and Design, IEEE, (1987).
38. C. Damas, B. Lambeau, A. van Lamsweerde: Scenarios, goals, and state machines: A win-win partnership for model synthesis, in Proceedings of the ACM SIGSOFT Symposium on Foundations of Software Engineering, pp. 197- 207, (2006).
39. I. Sommerville, P. Sawyer: Requirements Engineering: A Good Practice Guide, Wiley, (2006).
40. I. Bray: An Introduction to Requirements Engineering, Addison Wesley, (2002).
41. C. Damas, B. Lambeau, P. Dupont, A. Lamsweerde: Generating annotated behavior models from end-user scenarios, IEEE Transactions On Software Engineering, 31(12), pp. 1056-1073, (2005).
42. E. Letier, J. Kramer, J. Magee, S. Uchitel: Monitoring and control in scenario-based requirements analysis. In Proceedings the 27th international conference on Software engineering, St. Louis, MO, USA, pp. 382-391, (2005).
43. K. Wiegers: Software Requirements, 2nd Edition, Microsoft Press, ISBN 0-7356-1879-8, (2003).
44. J. Siddiqi, I. Morrey, R. Hibberd, G. Buckberry: Towards a System for the Construction, Clarification, Discovery and Formalization of Requirements, proceedings of first international conference on Requirements Engineering, IEEE, pp.230-238, (1994).

45. O. Laitenberger: Survey of Software Inspection Technologies, Handbook on Software Engineering and Knowledge Engineering, Citeseer, (2002).
46. R. Cleaveland, S. A. Smolka, S.T. Sims: An instrumentation-based approach to controller model validation, In Automotive Software Workshop, pag. 84–97. Springer, (2006).
47. I. Sommerville, P. Sawyer: Viewpoints: Principles, Problems and a Practical Approach to Requirements Engineering, Technical Report Ref: CSEG/15/1997, Lancaster University (1997).
48. J.C. Sampaio do Prado Leite, P. A. Freeman: Requirements Validation Through Viewpoint Resolution, IEEE transactions on Software Engineering, Vol. 17, No. 12, (1991).
49. T. Ambreen, N. Ikra, M. Usman: Empirical research in requirements engineering: trends and opportunities. Requirements Eng 23, 63–95, (2018).
50. A. Kokune, M. Mizuno, K. Kadoya, S. Yamamoto: FBCM: Strategy modeling method for the validation of software requirements, Journal of Systems and Software, Volume 80, Issue 3, Pages 314-327, (2007).
51. R. Machado, K. Lassen, S. Oliveira, M. Couto, P. Pinto: Requirements Validation: Execution of UML Models with CPN Tools, (2007).
52. D. Li, X. Li, J. Liu, Z. Liu: Validation of requirement models by automatic prototyping, Innov. Syst. Softw. Eng., vol. 4, no. 3, pp. 241–248, (2008).
53. R. Cavada, a. Cimatti, a. Mariotti, C. Mattarei, a. Micheli, S. Mover, M. Pensallorto, M. Roveri, a. Susi, and S. Tonetta: Supporting requirements validation: The EuRailCheck tool, ASE2009 - 24th IEEE/ACM Int. Conf. Autom. Softw. Eng., pp. 665–667, (2009).
54. I. Jureta, J. Mylopoulos, S. Faulkner: Analysis of multi-party agreement in requirements validation, 17th IEEE International Requirements Engineering Conference, pp. 57–66, (2009).
55. A. Cimatti, M. Roveri, A. Susi, S. Tonetta: From Informal Requirements to Property-Driven Formal Validation. In: Cofer D., Fantechi A. (eds) Formal Methods for Industrial Critical Systems. FMICS 2008. Lecture Notes in Computer Science, vol 5596. Springer, Berlin, Heidelberg, (2009).
56. D. Aceituna, H. Do, S. Lee: SQ^(2)E: An Approach to Requirements Validation with Scenario Question, Asia Pacific Software Engineering Conference, pp. 33-42, (2010).
57. L. Kof, R. Gacitua, M. Rouncefield, P. Sawyer: Ontology and Model Alignment as a Means for Requirements Validation. IEEE Fourth International Conference on Semantic Computing, 2010, pp. 46-51, (2010).
58. M. D. Sourour, N. Zarour: A methodology of Collaborative Requirements Validation in a cooperative environment, 10th International Symposium on Programming and Systems, Algiers, Algeria, pp. 140-147, (2011).
59. M. Kamalrudin, J. Grundy: Generating essential user interface prototypes to validate requirements, 26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011), pp. 564–567, (2011).
60. J. Holtmann, J. Meyer, M. von Detten: Automatic Validation and Correction of Formalized, Textual Requirements, IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops, pp. 486–495, (2011).
61. D. Aceituna, H. Do, S. W. Lee: Interactive requirements validation for reactive systems through virtual requirements prototype, Model-Driven Requirements Engineering Workshop, MoDRE 2011, pp. 1–10, (2011).
62. R. Sharma, K. K. Biswas: Using norm analysis patterns for automated requirements validation, Second IEEE International Workshop on Requirements Patterns (RePa), pp. 23–28, (2012).

63. A. Cimatti, M. Roveri, A. Susi, S. Tonetta: Validation of requirements for hybrid systems, ACM Trans. Softw. Eng. Methodol, vol. 21, no. 4, pp. 1–34, (2012).
64. M. Felderer, A. Beer: Using defect taxonomies for requirements validation in industrial projects, 21st IEEE International Requirements Engineering Conference, RE 2013 - Proceedings, pp. 296–301, (2013).
65. V. Gaur, A. Soni: A fuzzy traceability vector model for requirements validation. Int. J. Comput. Appl. Technol. 47, 2/3, 172–188, (2013).
66. Y. K. Lee, H. P. In, R. Kazman: Customer Requirements Validation Method Based on Mental Models, 21st Asia-Pacific Software Engineering Conference, vol. 1, pp. 199–206, (2014).
67. A. Rodrigues: Quality of Requirements Specifications - A Framework for Automatic Validation of Requirements, in Proceedings of the 16th International Conference on Enterprise Information Systems, pp. 96–107, (2014).
68. S. Nazir: A process improvement in requirement verification and validation using ontology, Asia-Pacific World Congress on Computer Science and Engineering, pp. 1-8, (2014).
69. S. Saito, J. Hagiwara, T. Yagasaki, K. Natsukawa: ReVAMP: Requirements Validation Approach using Models and Prototyping, Practical Cases of Requirements Engineering in End-User Computing, J. Inf. Process., vol. 23, no. 4, pp. 411–419, (2015).
70. N. Ali, R. Lai: A method of software requirements specification and validation for global software development, Requir. Eng., pp. 1–24, (2015).
71. S. Zafar, N. Farooq-Khan, M. Ahmed: Requirements simulation for early validation using Behavior Trees and Datalog, Inf. Softw. Technol., vol. 61, pp. 52–70, (2015).
72. W. Miao, G. Pu, Y. Yao, T. Su, D. Bao, Y. Liu, S. Chen, K. Xiong: Automated Requirements Validation for ATP Software via Specification Review and Testing, in Lecture Notes in Computer Science, vol 10009, pp. 26–40, (2016).
73. N. Yusop, M. Kamalrudin, S. Sidek, J. Grundy: Automated Support to Capture and Validate Security Requirements for Mobile Apps, Communications in Computer and Information Science, vol 671, pp. 97–112, (2016).
74. M. El-Attar, H. A. Abdul-Ghani: Using security robustness analysis for early-stage validation of functional security requirements, Requir. Eng., vol. 21, no. 1, pp. 1–27, (2016).
75. A. Moketar, M. Kamalrudin, S. Sidek, M. Robinson, J. Grundy: An automated collaborative requirements engineering tool for better validation of requirements, (2016).
76. B. Rosadini: Using NLP to Detect Requirements Defects: An Industrial Experience, Railway Domain, Grünbacher P., Perini A. (eds) Requirements Engineering: Foundation for Software Quality, Lecture Notes in Computer Science, vol 10153. Springer, Cham, (2017).
77. C. Buchholz, T. Vorsatz, S. Kind, R. Stark: SHPbench – A Smart Hybrid Prototyping Based Environment for Early Testing, Verification and (user based) Validation of Advanced Driver Assistant Systems of Cars, Procedia CIRP, Volume 60, Pages 139-144, (2017).
78. E. Stachtiari, A. Mavridou, P. Katsaros, S. Bliudze, J. Sifakis: Early validation of system requirements and design through correctness-by-construction, Journal of Systems and Software, Volume 145, Pages 52-78, (2018).
79. A. Chahin, K. Paetzold: Planning Validation & Verification Steps According to the Dependency of Requirements and Product Architecture, IEEE International Conference on Engineering, Technology and Innovation (ICE/ITMC), pp. 1-6, (2018).
80. K. Gruber, J. Huemer, A. Zimmermann, R. Maschotta: Automotive Requirements Validation and Traceability Analysis with AQL Queries, IEEE International Systems Engineering Symposium (ISSE), pp. 1-7, (2018).

81. S. Bayona-Oré, J. Chamilco, D. Perez: Software Process Improvement: Requirements Management, Verification and Validation, 14th Iberian Conference on Information Systems and Technologies (CISTI), pp. 1-5, (2019).
82. Y. Yang, W. Ke, X. Li: RM2PT: Requirements Validation through Automatic Prototyping, IEEE 27th International Requirements Engineering Conference (RE), págs. 484-485, (2019).
83. I. Atoum: A Scalable Operational Framework for Requirements Validation Using Semantic and Functional Models, In Proceedings of the 2nd International Conference on Software Engineering and Information Management (ICSIM 2019). Association for Computing Machinery, New York, NY, USA, 1–6, (2019).
84. S. F. Alshareef, A. M. Maatuk, T. M. Abdelaziz, Mohammed Hagal: Validation Framework for Aspectual Requirements Engineering (ValFAR). In Proceedings of the 6th International Conference on Engineering & MIS (ICEMIS'20). Association for Computing Machinery, New York, NY, USA, Article 42, 1–7, (2020).
85. D. Iqbal, A. Abbas, M. Ali, M. U. S. Khan, R. Nawaz: Requirement Validation for Embedded Systems in Automotive Industry Through Modeling, IEEE, vol. 8, pp. 8697-8719, (2020).
86. X. Fei, C. Bin, Z. Siming: A Methodology of Requirements Validation for Aviation System Development, Chinese Control And Decision Conference (CCDC), págs. 4484-4489, (2020).
87. A. Mashkoor, M. Leuschel, A. Egyed: Validation Obligations: A Novel Approach to Check Compliance between Requirements and their Formal Specification, (2021).
88. IEEE Recommended Practice for Software Requirements Specifications, IEEE Std 830-1998 vol., no., pp.1-40, (1998).
89. J. N. Martin: Overview of the EIA 632 standard: processes for engineering a system, 17th DASC. AIAA/IEEE/SAE. Digital Avionics Systems Conference. Proceedings (Cat. No.98CH36267), pp. B32-1, (1998).
90. SAE international Group. APR4754A Guideline for Development of Civil Aircraft and Systems. vol. 4970, (2011).
91. S. Kovacevic: UML and User Interface Modeling, The Unified Modeling Language. UML'98: Beyond the Notation, pp. 514-514, (1999).
92. S. Maalem, N. Zarour: Challenge of validation in requirements engineering. Journal of Innovation in Digital Ecosystems, 3(1):15–21, (2016).

Anexos

Anexo 1

Ref. Art.	Quién	Qué	Cuando	Porqué	Cómo
[50]		Presenta un enfoque que evalúa metas y objetivos de la organización con modelos de estrategia, observaciones de campo, análisis de datos comerciales y el mapa de estrategia de colaboración.	Etapa de planificación, análisis y especificación	Evaluar la integridad de las metas y objetivos fundamentales de la organización que se utilizan como requerimientos para el desarrollo del sistema.	Técnicas: Árbol de análisis de objetivos, tarjetas de observaciones de campo, mapa de estrategias y matriz de colaboración. Controles: observaciones de campo y métodos de análisis de datos de KPI.

Ref. Art.	Quién	Qué	Cuando	Porqué	Cómo
[51]	Usuarios Desarrollador	Presenta un enfoque basado en prototipos interactivos para validar los requerimientos del flujo de trabajo.	Etapa de elicitacion y análisis	Identificar defectos e incoherencias en los requerimientos.	Herramientas: CPN Tools BRITNeY Animation Técnicas: prototipos, animación. Lenguaje de Modelado Unificado (UML)

Ref. Art.	Quién	Qué	Cuando	Porqué	Cómo
[52]		Presenta un enfoque que convierte un modelo re requerimientos en prototipos ejecutables.	Etapa de análisis y diseño.	Detectar errores en los requerimientos del sistema.	Herramientas: AutoPA3.0 Técnicas: prototipos del modelo de requerimientos. Lenguaje de Modelado Unificado (UML)

Ref. Art.	Quién	Qué	Cuando	Porqué	Cómo
[53]	Usuarios	Presenta una herramienta que analiza los requerimientos en lenguaje natural para categorizarlos y estructurarlos	Etapa de especifica-ción	Verificar la consistencia formalmente, la ausencia de contradiccio-nes lógicas en el documento de requerimientos	Herramientas: EuRailCheck Técnicas: verificación del modelo de requerimientos. Lenguaje de Modelado Unificado (UML)

Ref. Art.	Quién	Qué	Cuando	Porqué	Cómo
[54]	Usuarios Ingeniero de requerí-mientos.	Presenta un enfoque que modela y analiza una discusión entre las partes interesadas sobre la especificación de requerimientos	Etapa de elicitación.	Obtener la validez y aceptabilidad del contenido de la especificación de requerimientos	Herramienta: Algoritmo para evaluar la condición de aceptabilidad. Lenguaje para la representación de acuerdos entre las partes.

Ref. Art.	Quién	Qué	Cuando	Porqué	Cómo
[55]		Presenta un enfoque que formaliza segmentos categorizados de requerimientos en lenguaje natural.	Etapa de especificación	Verificar la consistencia de la especificación de requerimientos	Herramienta: IBM Rational RequisitePro Diagrama de clases, diagrama de secuencias, máquinas de estado (UML) y lenguaje natural controlado (CNL).

Ref. Art.	Quién	Qué	Cuando	Porqué	Cómo
[56]		Presenta un enfoque que desarrolla un modelo a partir de la especificación de requerimientos	Etapa de especificación	Validar y mejorar el documento de requerimientos	Modelo basado en escenarios. Técnicas: revisión/inspección

Ref. Art.	Quién	Qué	Cuando	Porqué	Cómo
[57]	Analistas	Presenta un enfoque que desarrolla modelos de comportamiento a partir de los textos de requerimientos	Etapa de especificación	Identificar la falta de información e inconsistencias en el documento de requerimientos	Modelo basado en ontologías de dominio genéricas. Técnicas de procesamiento de lenguaje natural

Ref. Art.	Quién	Qué	Cuando	Porqué	Cómo
[58]	Analistas Equipo ER	Presenta un enfoque colaborativo y distribuido basado en los conceptos de competencias y multipuntos de vista	Etapa de especifica-ción	Verificar las cualidades básicas en el documento de requerimientos	Técnicas: Listas de verificación, orientado al punto de vista, inspección, revisión y prototipos. Estándar IEEE 830 [88] Control: uso de métricas.

Ref. Art.	Quién	Qué	Cuando	Porqué	Cómo
[59]		Presenta un enfoque que modela sistemas técnicos automotrices basados en requerimientos de consultas para soporte.	Etapa de elicitacion y análisis	Asegurar la trazabilidad y gestionar la coherencia de la especificación de requerimientos	Herramienta: MaramaAI Técnica: Prototipo Lenguaje de Modelado Unificado (UML) Lenguaje Acceleo Query Language (AQL)

Ref. Art.	Quién	Qué	Cuando	Porqué	Cómo
[60]	Ingeniero de requerí-mientos	Presenta un enfoque que desarrolla un metamodelo de requerimientos textuales.	Etapa de especifica-ción	Identificar inconsisten-cias y requerimientos incompletos de la especificación. de requerimientos	Lenguaje natural controlado (CNL) Patrones estructurales y operaciones de corrección.

Ref. Art.	Quién	Qué	Cuando	Porqué	Cómo
[61]		Presenta un prototipo virtual de requerimientos con la participación de las partes interesadas.	Etapa de elicitacion y análisis	Minimizar los errores en los requerimientos .	Técnica: Prototipo virtual de requerimientos (VRP)

Ref. Art.	Quién	Qué	Cuando	Porqué	Cómo
[62]		Presenta un enfoque que genera patrones de comportamien to de requerimientos basados en normas de la organización.	Etapa de elicitacion y especifica-ción	Identificar la completitud, inconsistencia y los inequívocos del documento de requerimientos	Técnicas de procesamiento del lenguaje natural (PNL). Patrones de análisis de normas desarrollados con escenarios.

Ref. Art.	Quién	Qué	Cuando	Porqué	Cómo
[63]	Experto en dominio	Presenta un enfoque que fragmenta y categoriza los requerimientos informales según su función.	Etapa de especifica-ción	Abordar problemas de conversión informal a formal, trazabilidad, automatiza-ción, aceptación del usuario y escalabilidad.	Lenguaje formal OTHELLO (Object Temporalwith Hybrid Expressions Linear-time LOGic) para sistemas híbridos en aplicaciones críticas para la seguridad

Ref. Art.	Quién	Qué	Cuando	Porqué	Cómo
[64]		Presenta un enfoque que integra las taxonomías por defecto a la validación de los requerimientos	Etapa de elicitacion, análisis y especifica-ción.	Identificar anomalías en el documento de requerimiento.	Técnicas: Revisión y prueba.

Ref. Art.	Quién	Qué	Cuando	Porqué	Cómo
[65]		Presenta un enfoque que desarrolla un modelo vectorial de trazabilidad difusa para requerimientos	Etapa de elicitacion y especifica-ción.	Asegurar la trazabilidad, exactitud, integridad y consistencia de requerimientos	Técnicas de trazabilidad de requerimientos. User Story Cards (USC) Agent Cards (AC) Control: uso de métricas.

Ref. Art.	Quién	Qué	Cuando	Porqué	Cómo
[66]		Presenta un enfoque que desarrolla modelos de los comportamien tos de clientes y sus estados mentales.	Etapa de especifica-ción	Identificar consistencia, completitud y exactitud del documento de requerimientos	Técnicas: revisión e inspección.

Ref. Art.	Quién	Qué	Cuando	Porqué	Cómo
[67]		Presenta un enfoque que desarrolla un modelo genérico para validar automáticame nte la especificación de requerimientos	Etapa de especifica-ción.	Identificar inconsisten-cias, incompletitud y ambigüedades del documento de requerimientos	Herramienta: SpecQuA (Requirements Specification Quality Analyzer) Técnica: Inspección basada en casos de prueba.

Ref. Art.	Quién	Qué	Cuando	Porqué	Cómo
[68]	Usuario Analista Ingeniero de software	Presenta un enfoque que desarrolla un modelo ontológico para mejorar el proceso de validación de requerimientos	Etapa de especifica-ción	Eliminar ambigüedad, incompletitud e inconsistencia en el documento de requerimientos	Técnicas: prototipos animación

Ref. Art.	Quién	Qué	Cuando	Porqué	Cómo
[69]		Presenta una herramienta que genera automáticame nte prototipos a partir de modelos de requerimientos	Etapa de elicitación	Detectar automática-mente la inconsistencia y la completitud de los requerimientos	Herramienta: RM2PT (Requirements Modeling and Automatic Prototyping) Técnicas: prototipo animación

Ref. Art.	Quién	Qué	Cuando	Porqué	Cómo
[70]	Ingeniero de requeri-mientos Analista	Presenta un enfoque que desarrolla un método de especificación y validación de requerimientos GSD (Global Software Development).	Etapa de elicitacion y especifica-ción	Lograr la aceptabilidad, no ambigüedad, integridad, verificabilidad y comprensibili-dad de la especificación de requerimientos	Gráficos de requerimientos por zona y huso horario. Matriz de validación de requerimientos.

Ref. Art.	Quién	Qué	Cuando	Porqué	Cómo
[71]		Presenta un enfoque que desarrolla un simulador que captura estados del sistema como reglas y hechos en una base de datos.	Etapa de especifica-ción	Identificar defectos en la especificación de requerimientos	Herramienta: SimTree Técnica: Simulación con árboles de comportamiento

Ref. Art.	Quién	Qué	Cuando	Porqué	Cómo
[72]		Presenta una herramienta que desarrolla un prototipo ejecutable basado en escenarios.	Etapa de análisis y especifica-ción	Facilitar la integridad y corrección en la especificación de requerimientos	Herramienta: VDM Tool (Vienna Development Method) Técnicas: inspección, revisión y pruebas de diagramas.

Ref. Art.	Quién	Qué	Cuando	Porqué	Cómo
[73]	Ingeniero de requeri-mientos.	Presenta una herramienta que captura y valida requerimientos de seguridad de las aplicaciones móviles.	Etapa de elicitación y especifica-ción	Garantizar la coherencia, integridad, y corrección de los requerimientos	Herramienta: MobiMEReq Técnica: pruebas basadas en modelos utilizando casos de uso.

Ref. Art.	Quién	Qué	Cuando	Porqué	Cómo
[74]		Presenta un enfoque que desarrolla un conjunto integral de pruebas de aceptación de seguridad basado en modelos de casos de uso.	Etapa de elicitación.	Garantizar la robustez de los requerimientos de seguridad.	Técnicas: análisis de robustez de seguridad. Pruebas de aceptación de seguridad.

Ref. Art.	Quién	Qué	Cuando	Porqué	Cómo
[75]	Cliente Ingeniero de requeri-mientos.	Presenta una herramienta de comunicación y colaboración automatizada.	Etapa de elicitacion.	Obtener requerimientos correctos y completos.	Herramienta: TestMEReq Técnica: casos de prueba. Historias de usuario o escenarios.

Ref. Art.	Quién	Qué	Cuando	Porqué	Cómo
[76]		Presenta un enfoque que genera patrones con técnicas de PNL (Procesamiento del Lenguaje Natural).	Etapa de especificación	Identificar defectos en el documento de requerimientos industriales.	Herramienta: GATE (Resuelve problemas de procesamiento de texto)

Ref. Art.	Quién	Qué	Cuando	Porqué	Cómo
[77]		Presenta un enfoque que desarrolla un entorno de creación de prototipos para manejar las demandas de ADAS (sistemas avanzados de asistencia al conductor)	Etapa de elicitacion.	Identificar defectos e incoherencias en los requerimientos	Técnicas: prototipos, casos de uso.

Ref. Art.	Quién	Qué	Cuando	Porqué	Cómo
[78]		Presenta un enfoque que desarrolla modelos para la formalización y validación de los requerimientos del sistema.	Etapa de especificación y diseño.	Detectar las inconsistencias y defectos de los requerimientos	Herramientas: DesignBIP edición grafica basada en la web. D-Finder analiza modelos BIP. Modelos y arquitecturas BIP (Behavior-Interaction-Priorities)

Ref. Art.	Quién	Qué	Cuando	Porqué	Cómo
[79]		Presenta un enfoque que desarrolla modelos de visualización de las descripciones de requerimientos y sus dependencias.	Etapa de especifica-ción.	Conseguir la trazabilidad de los requerimientos	Modelo basado en redes y grafos.

Ref. Art.	Quién	Qué	Cuando	Porqué	Cómo
[80]		Presenta un enfoque que desarrolla modelos para sistemas técnicos de automóviles y consultas de análisis.	Etapas del modelado del sistemas	Identificar errores y analizar la trazabilidad de los requerimientos en el documento de especificación.	Modelo basado en UML (Unified Modeling Language)

Ref. Art.	Quién	Qué	Cuando	Porqué	Cómo
[81]		Presenta un enfoque que presenta buenas practicas CMMI para el proceso de gestión de requerimientos	Ciclo de vida del proceso.	Identificar inconsisten-cias y defectos de los requerimientos	Modelo IDEAL (Initiating, Diagnosing, Establishing, Acting & Learning) Modelo SCAMPI(Standard CMMI Appraisal Method for Process Improvement)

Ref. Art.	Quién	Qué	Cuando	Porqué	Cómo
[82]		Presenta una herramienta para generar prototipos a partir de modelos de requerimientos	Etapa de análisis y diseño.	Mejorar la inconsistencia de los requerimientos	Herramienta: RM2PT Técnicas: prototipos, casos de uso.

Ref. Art.	Quién	Qué	Cuando	Porqué	Cómo
[83]	Ingeniero de software Cliente	Presenta un enfoque que define un marco para prender, predecir, y reconocer los defectos de los requerimientos	Etapa de elicitación	Detectar la inconsistencia de los requerimientos	Lenguaje IDEF (Integration DEFinition) Técnica: aprendizaje automático. Modelos de similitud semántica.

Ref. Art.	Quién	Qué	Cuando	Porqué	Cómo
[84]	Desarrolla-dor Cliente	Presenta un enfoque que presenta un marco para identificar y formalizar aspectos de los requerimientos	Etapa de elicitación	Evaluar la consistencia y completitud de los requerimientos	Técnicas: Lista de verificación. Modelado con diagrama de secuencia.

Ref. Art.	Quién	Qué	Cuando	Porqué	Cómo
[85]		Presenta un enfoque que desarrolla un modelo mediante la generación de códigos y la simulación.	Etapas del modelado del sistema integrado.	Eliminar ambigüedades e inconsisten-cias de los requerimientos	Técnicas: prototipos pruebas. Modelado con redes de Petri.

Ref. Art.	Quién	Qué	Cuando	Porqué	Cómo
[86]		Presenta un enfoque que propone una metodología de validación de requerimientos para el desarrollo de sistemas de aviación	Ciclo de vida del desarrollo del producto.	Evaluar la exactitud, integridad y lo completo de los requerimientos	Técnicas: Listas de verificación, pruebas, inspección y revisión. Estándar EIA 632 [89] Certificación ARP4754A [90]

Ref. Art.	Quién	Qué	Cuando	Porqué	Cómo
[87]		Presenta un enfoque que desarrolla un prototipo capaz de diseñar y validar escenarios.	Etapa de especifica-ción	Detectar errores en el documento de requerimientos	Técnicas: animación, simulación, pruebas.