Índice Capítulo 1

Capítulo 1. Fundamentos Teóricos

1.1 Ingeniería de requisitos en el proceso de desarrollo de software

La industria del software presenta índices de éxito en los proyectos relativamente bajos si se tienen en cuenta estudios e investigaciones realizados por Standish Group's en los reportes Chaos, los cuales analizan el comportamiento de proyectos desde el año 1994. Según estos estudios aproximadamente el 50% de los expertos encuestados considera que el comportamiento de fallo en los proyectos es idéntico al de hace 10 años, el 48% considera que existen más fallos en los proyectos hoy en día y solamente el 2% considera que ha existido una mejoría en este aspecto. Los datos estadísticos reflejan que en el año 2018 solo el 16.2% de los proyectos de software fueron completados cumpliendo con el tiempo, el presupuesto y con al menos la mitad de los requisitos planteados inicialmente. Estos datos son incluso peores teniendo en cuenta los proyectos llevados a cabo por grandes empresas donde estos datos pueden llegar al 9% [1, 2].

Las principales causas de fallo de un proyecto han sido descritas en muchos estudios entre las principales se encuentran:

- Incompleta o errónea declaración de requisitos.
- Requisitos cambiantes.
- Expectativas poco realistas.
- Poca claridad en los objetivos y metas.
- Incompetencia de la tecnología.
- Metas de tiempo poco realistas.
- Calidad del producto.
- Poca claridad de los procesos de negocio.
- Satisfacción del cliente.
- Poca comunicación entre clientes, desarrolladores y usuarios finales.
- Políticas de los stakeholders (partes interesadas).

Como se observa en los estudios el porcentaje de fallos en este proceso sigue siendo considerablemente alto y el costo de corregir un error a partir de un mal proceso de IR (Ingeniería de requisitos) puede suponer hasta 5 veces superior, en comparación a otros

errores, aspectos que motivan y fundamentan la necesidad de continuar con el estudio de esta rama con el objetivo de disminuir los porcientos de fallos esta temprana etapa del proyecto. [3, 4].

Somerville [5], plantea: “Los requerimientos para un sistema son descripciones de lo que el sistema debe hacer: el servicio que ofrece y las restricciones en su operación. Tales requerimientos reflejan las necesidades de los clientes por un sistema que atienda cierto propósito, como sería controlar un dispositivo, colocar un pedido o buscar información. Al proceso de descubrir, analizar, documentar y verificar estos servicios y restricciones se le llama ingeniería de requisitos (IR)”.

Por otra parte, Pressman [6] define la IR como un puente entre el diseño y la construcción, argumentando que existen diferentes criterios de cuál es el comienzo de este puente. Una parte de los expertos plantea que inicia en los pies de los participantes en el proyecto (por ejemplo, gerentes, clientes y usuarios), donde se definen las necesidades del negocio, se describen los escenarios de uso, se delinean las funciones y características y se identifican las restricciones del proyecto. Otros sugieren que empieza con una definición más amplia del sistema, donde el software no es más que un componente del dominio del sistema mayor. Pero sin importar el punto de arranque, el recorrido por el puente lo lleva a uno muy alto sobre el proyecto, lo que le permite examinar el contexto del trabajo de software que debe realizarse; las necesidades específicas que deben abordar el diseño y la construcción; las prioridades que guían el orden en el que se efectúa el trabajo, y la información, las funciones y los comportamientos que tendrán un profundo efecto en el diseño resultante.

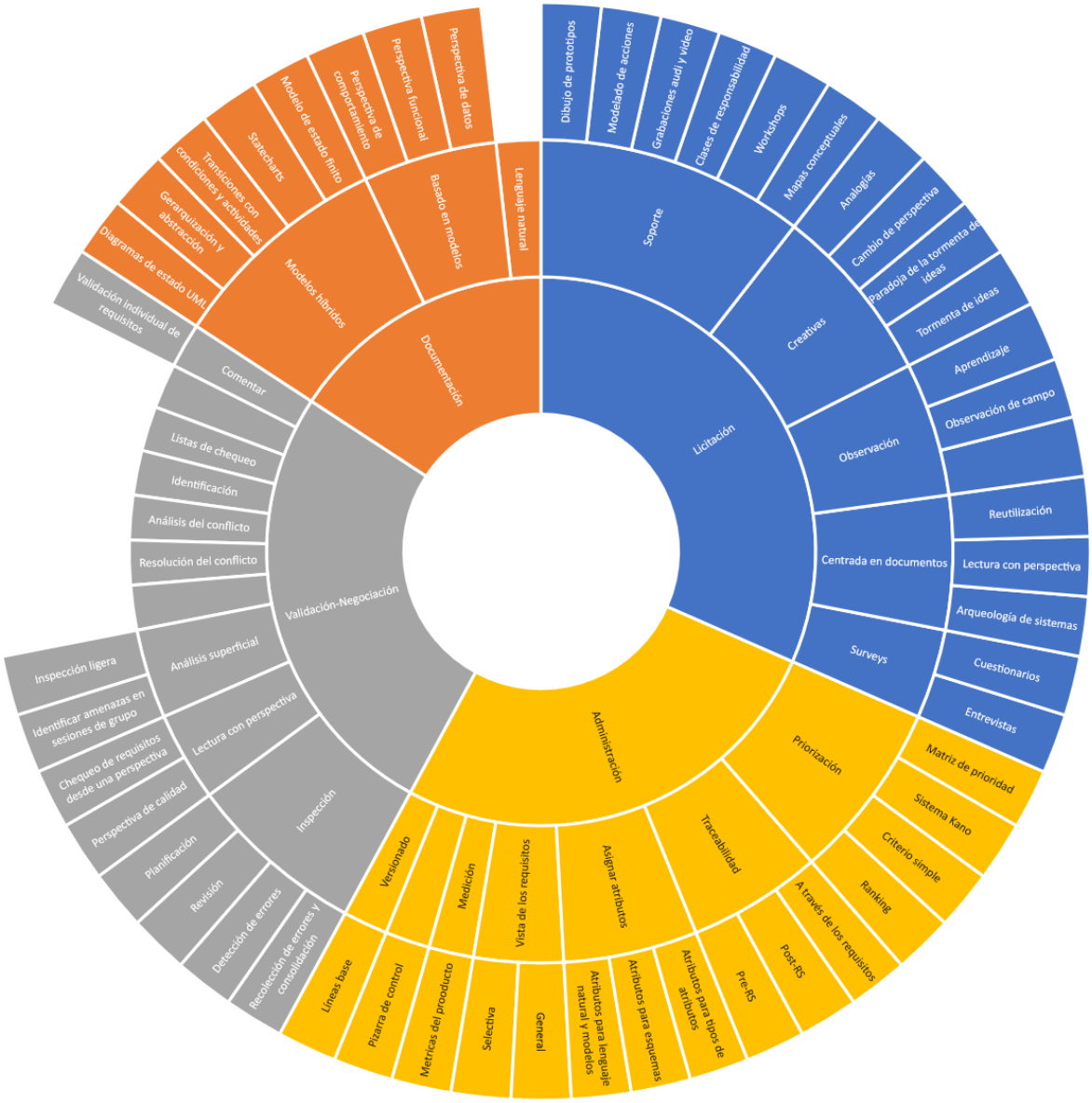
La ingeniería de requisitos proporciona el mecanismo apropiado para entender lo que desea el cliente, analizar las necesidades, evaluar la factibilidad, negociar una solución razonable, especificar la solución sin ambigüedades, validar la especificación y administrar los requerimientos a medida que se transforman en un sistema funcional. Incluye según Pressman siete tareas diferentes: concepción, indagación, elaboración, negociación, especificación, validación y administración. Es importante destacar que algunas de estas tareas ocurren en paralelo y que todas se adaptan a las necesidades del proyecto [6].

Se puede concretar que un requisito de software es:

1. Condición o capacidad necesitada por un usuario para resolver un problema o alcanzar un objetivo.

2. Condición o capacidad que, solucionada o procesada por un sistema, un componente de sistema para satisfacer contratos, estándares, especificaciones u otros documentos formales.
3. Representación documental de una capacidad o condición como se manifiesta en los puntos 1 y 2.

El proceso de IR se puede dividir en cuarto actividades centrales, ***Licitación, Documentación, Validación-Negociación, Administración***. Algunos autores plantean variantes de estas, pero fundamentalmente estas 4 actividades componen el núcleo del proceso [7].



La comunicación es un aspecto vital en los requisitos de software, para su entendimiento, aceptación y procesamiento. Es necesario la selección de un medio común de comunicación debido a las dificultades que pueden implicar distintos factores como:

- Deficiencia en la comunicación en lenguaje natural
- Diferencias culturales y educacionales
- Interpretaciones
- Comprensión o dominio del contexto en que se trabaja
- Niveles de redundancia en la comunicación verbal
- Diferencia en los idiomas

En adición a los problemas que pueda traer el desconocimiento de los dominios del lenguaje que se trata, otro problema puede ser que la información puede no ser transmitida correctamente causada por problemas de percepción humana.

La especificación de un requisito puede ser muy variada, de forma general estos pueden ser declarados en lenguaje natural, lenguajes formales, modelos o diagramas (utilizando lenguajes descriptivos como UML), así como otros formatos en dependencia de los métodos o metodologías utilizados para el proceso de desarrollo. Generalmente la construcción de estos artefactos o especificaciones a partir de la captura de información inicial resulta una de las etapas críticas en la concepción de los requisitos. En adición este es un proceso generalmente cooperativo donde interactúan varios actores, lo cual agrega un nivel de dificultad a la hora de la organización, estandarización y gestión de requisitos de un proyecto.

Existen diversas formas de clasificación de los requisitos como las propuestas por CMMI [8] o SPICE [9]. Pero las más comúnmente usadas son:

1. Requisito funcional: Enfocado al comportamiento que debe seguir o que debe cumplir una función del sistema.
2. Requisito de calidad: Corresponde a los aspectos de calidad del sistema que no son cubiertos por los requisitos funcionales, estos son frecuentemente clasificados como no funcionales.
3. Restricciones: Requisito que limita la solución en el espacio y el contexto más allá de lo marcado por los requisitos funcionales y de calidad.

El proceso de captura de información, aunque aún mantiene bases en aspectos tradicionales como las entrevistas y encuestas ha sufrido también de una evolución. Sobre todo, marcado por los diferentes nuevos contextos a los que se ha enfrentado la industria del desarrollo de sistemas. El trabajo a distancia, las entrevistas no presenciales a través de audio, video o videollamadas, el aumento en la complejidad de los negocios y problemas a resolver, uso de correo electrónico, las observaciones técnicas, la participación de varios especialistas de diferentes ramas, entre otras variantes. Influyendo lo anterior directamente en el aumento de la complejidad de este proceso, su procesamiento, y en muchas ocasiones en elevados volúmenes de información y del tiempo a utilizar para su procesamiento.

Para el proceso de IR se pueden definir tres grupos fundamentales como fuentes de información:

1. Stakeholders: Personas u organizaciones que de forma directa o indirecta tienen influencia sobre los requisitos de un sistema. Estos pueden ser usuarios finales, operadores, clientes, desarrolladores, arquitectos y probadores.
2. Documentos: Una de las principales fuentes de información para obtener requisitos de software. Estos pueden ser, documentos de organizaciones, manuales técnicos, contratos, documentos legales, reportes de errores, observaciones técnicas. En cuento a los formatos de estos no existen normas que restrinjan u organicen estos documentos en una estructura determinada.
3. Sistemas en operación, que puedan ser predecesores de las soluciones que se desean obtener. De soluciones precedentes se pueden obtener soluciones a partir de requisitos comunes.

Ilustración 1 Fuentes de información

1.1.1 Estándares y formalizaciones de requisitos de software

Para la documentación de requisitos de software se puede estructurar en tres grupos fundamentales:

1. Utilizando lenguaje natural: Esta es la variante más comúnmente utilizada en la práctica, teniendo la ventaja de que los stakeholders no deberán aprender nuevas notaciones, y que el lenguaje puede utilizarse de múltiples propósitos, desde estructuras formales hasta descripciones detalladas. En cambio, los requisitos en lenguaje natural pueden ser ambiguos, y perder perspectiva si no se redactan correctamente.
2. Modelos conceptuales: A diferencia del lenguaje natural los diferentes modelos conceptuales no pueden utilizarse de forma universal. Al documentar requisitos mediante modelos se deben utilizar lenguajes de modelado específicos, que deben ser utilizados de forma correcta para poder alcanzar el objetivo deseado. Si el lenguaje de modelado es correctamente utilizado, garantiza que el requisito responde a una perspectiva determinada sin ambigüedades. Esta formalización permite documentos más reducidos y sencillos de comprender para especialistas preparados. Entre los modelos más utilizados se encuentran:

- Diagramas de clase
- Diagramas de casos de uso
- Diagramas de actividades
- Diagramas de estado

3. Documentos híbridos: Actualmente la variante que más fuerza va tomando. La utilización combinada de los dos puntos anteriores permite elaborar un documento enfocado a la audiencia objetivo. Brindando diferentes perspectivas del sistema. La utilización combinada del lenguaje natural y modelos conceptuales arrastra las desventajas de ambas opciones, pero se complementa por la combinación de las fortalezas y su versatilidad.

Para determinar que estructuras de requisitos utilizar, los estereotipos adecuados en cada situación existen diferentes estándares desde los métodos tradicionales hasta las más actuales metodologías ágiles de desarrollo. Entre los estándares más utilizados se encuentran:

1. ***Rational Unified Process (RUP)*** [10]*:* Este proceso es usualmente utilizado al desarrollar bajo el paradigma orientado a objeto. En este se utilizan modelos para describir el negocio y la solución mediante el uso de diferentes artefactos como: Reglas del negocio, casos de uso, combinando el lenguaje natural con modelos conceptuales. La especificación de requisitos normalmente es redactada utilizando la estructura *software requirements specification (SRS),* la cual está muy relacionada al estándar ISO/IEC/IEEE 29148:2011.

   Entre los modelos que propone RUP se encuentran:

   - Modelos de casos de uso
   - Modelo de análisis
   - Modelo de diseño
   - Modelo de despliegue
   - Modelo de implementación
   - Modelo de prueba

Existen dependencias entre muchos de los modelos. Como ejemplo, se indican las dependencias entre el modelo de casos de uso y los demás modelos.

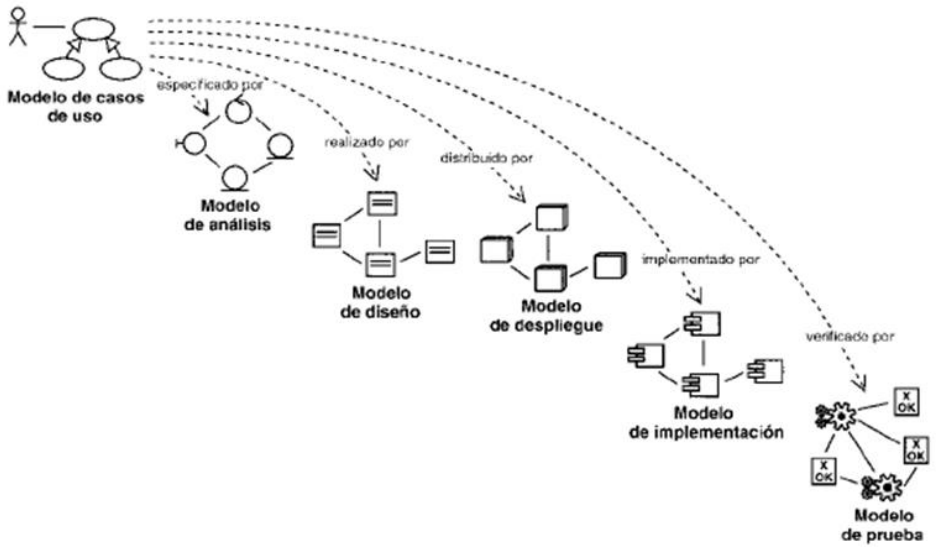


Ilustración 2 Dependencias entre los modelos RUP

2. **Estándar ISO/IEC/IEEE 29148:2011** [11]: Contiene normas establecidas para la documentación de requisitos de software. El estándar sugiere dividir el documento de especificación de requisitos en 5 capítulos utilizando el lenguaje natural como variante a utilizar.
    - Información introductoria
    - Referencias
    - Requisitos del sistema (funcionales, rendimiento, interfaces)
    - Medidas para la verificación
    - Apéndice (Se aclaran los elementos que se asumen, identifican las dependencias)
3. **Modelo V**: Define diferentes estructuras a partir de quien es el creador:
    - Especificación de requisitos del cliente: Elaborado por el cliente, incluye servicios, restricciones, explicación del proceso. Generalmente se escribe para responder el **qué** se hace y **para qué** se hace.
    - Especificación de requisitos del sistema: Se basa en la especificación del cliente, pero tiene detalles del diseño e implementación que elabora el contratista. Se elabora a partir del proceso y restricciones establecidas por el cliente.
4. **CMMi** [12]: Este estándar plantea como propósito del desarrollo de requisitos de software, producir y analizar los requerimientos de cliente, de producto y de componente del producto. Aplicando un conjunto de metas y prácticas específicas:

SG 1 Desarrollar los requerimientos de cliente.

SP 1.1 Obtener las necesidades.

SP 1.2 Desarrollar los requerimientos de cliente.

SG 2 Desarrollar los requerimientos de producto.

SP 2.1 Establecer los requerimientos de producto y de componentes del producto.

SP 2.2 Asignar los requerimientos de componentes del producto.

SP 2.3 Identificar los requerimientos de interfaz.

SG 3 Analizar y validar los requerimientos.

SP 3.1 Establecer los conceptos operativos y los escenarios.

SP 3.2 Establecer una definición de la funcionalidad requerida.

SP 3.3 Analizar los requerimientos.

SP 3.4 Analizar los requerimientos para alcanzar el equilibrio.

SP 3.5 Validar los requerimientos.

Como pare del proceso de IR CMMi plantea las bases para la gestión de os requisitos a partir de gestionar los requerimientos de los productos y de los componentes del producto del proyecto, e identificar inconsistencias entre esos requerimientos y los planes y productos de trabajo del proyecto.

Planteando las siguientes metas y prácticas:

SG 1 Gestionar los requerimientos.

SP 1.1 Obtener una comprensión de los requerimientos.

SP 1.2 Obtener el compromiso sobre los requerimientos.

SP 1.3 Gestionar los cambios de los requerimientos.

SP 1.4 Mantener la trazabilidad bidireccional de los requerimientos.

SP 1.5 Identificar las inconsistencias entre el trabajo del proyecto y los requerimientos.

5. Estándar IEEE 830: El estándar IEEE 830-1998 está enfocado en recomendaciones prácticas para la especificación de requerimientos, fue desarrollado por la IEEE y la IEEE-SA (Standards Association), indica la estructura y organización de toda la información que debe incluirse en un buen documento de especificación de requerimientos de software. Los objetivos que tiene este estándar son: ayudar a los clientes a describir con precisión lo que quieren en el software y a las personas encargadas de recibir esta información establecer una estructura estándar (definir el formato y contenido de las especificaciones de requerimientos de software y manual del mismo) para la especificación de requerimientos de software (ERS) en

sus organizaciones. La especificación de requerimientos de software obliga a los involucrados en el desarrollo del software a considerar todos los requerimientos de forma rigurosa antes de iniciar el diseño y codificación del mismo, con la finalidad de evitar el rediseño, proporcionando las bases necesarias para la estimación de tiempo y costo, referencias de verificación y validación.

Una de las características más distinguibles del estándar IEEE 830 es que los requisitos deben ser escritos de la siguiente manera: “El sistema debe…”.

6. Metodologías ágiles e historias de usuario: Las Historias de Usuario son un enfoque de requerimientos ágil que se focaliza en establecer conversaciones acerca de las necesidades de los clientes. Son **descripciones cortas y simples** de las funcionalidades del sistema, narradas desde la perspectiva de la persona que desea dicha funcionalidad, usualmente un usuario. Las historias de usuario enfatizan la comunicación verbal por sobre la escrita: la desventaja de documentar los requerimientos de software es que los clientes pueden obtener por resultado lo que el equipo de desarrollo ha interpretado, y no precisamente lo que el cliente necesita. En las metodologías ágiles, el objetivo es documentar lo menos posible en las historias de usuario, se escribe lo estrictamente necesario para recordar que deben establecerse las conversaciones con los clientes para definir los detalles de la implementación.

1.2 Rol de lenguaje natural en los requisitos de software

El lenguaje natural juega un papel fundamental en todas las actividades de la ingeniería de requisitos, pero fundamentalmente en la etapa de licitación. Su principal ventaja es la capacidad de describir cualquier circunstancia de forma universal. Pero este posee la característica de ser generalmente ambiguo, lo cual dificulta satisfacer la condición de los requisitos de no ambigüedad. Este proceso está influenciado por la percepción y transformación del conocimiento [13, 14, 15].

Ilustración 3 Transformación de la información

Esta transformación posee varias etapas vitales en la construcción de un requisito:

- Nominalización: Consiste en la reducción del proceso, este se debe acotar excluyendo elementos innecesarios en su descripción sin dejar de definir el proceso en su totalidad.
- Términos sin índice de referencia: no deben existir términos que no estén definidos su significado claramente.
- Cuantificadores universales: Se deben especificar cantidades y frecuencias.
- Estructuras de condición: El no identificar estructuras de condición generalmente significa errores en los requisitos, siempre se debe especificar el comportamiento al ocurrir un evento o al no ocurrir este.
- Verbos de procesos: Algunos verbos de acciones no se consideran completos sin complementos.

1.2.1 Construcción de un requisito en lenguaje natural.

Para la construcción de un requisito en lenguaje natural se pueden plantear 5 pasos fundamentales. Con el objetivo de garantizar la calidad de los requisitos mediante el uso de una estructura definida y glosarios de términos.

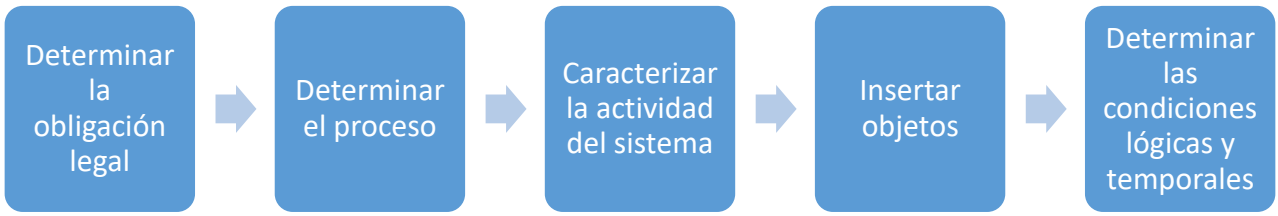


Ilustración 4 Proceso de construcción de un requisito

Obteniendo la siguiente estructura como base para la construcción del requisito en lenguaje natural:

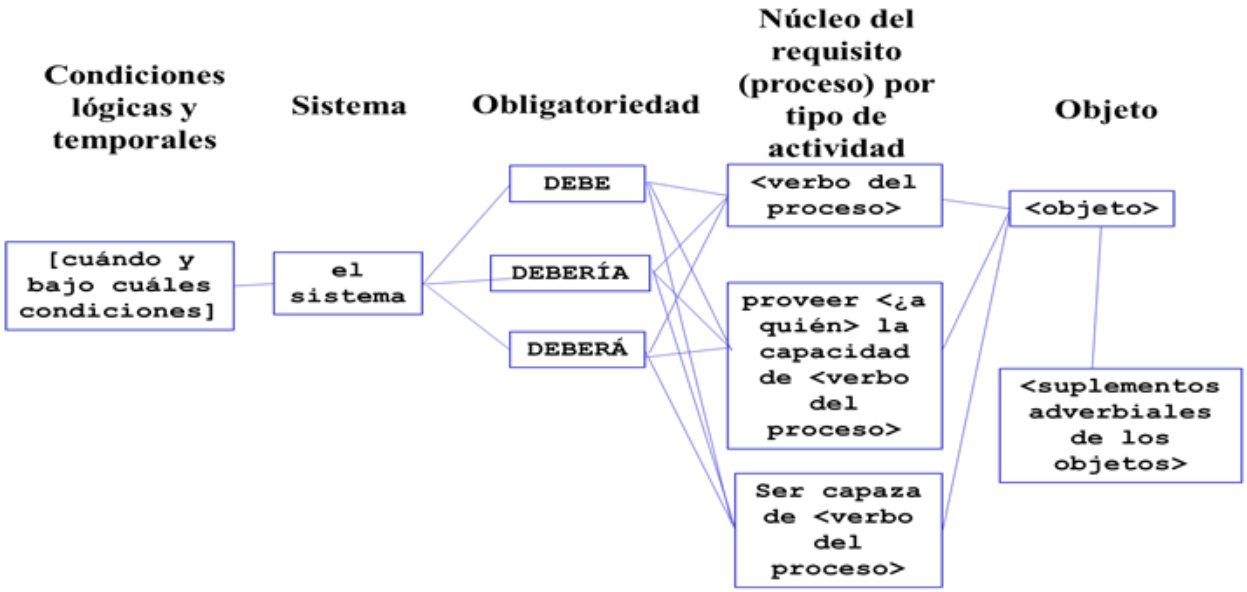


Ilustración 5 Estructura de un requisito en lenguaje natural

1.2.2 Utilización de modelos

La documentación de requisitos basada en modelos es utilizada para tres tipos de requisitos [7].

1. Metas: Describe las intenciones de los stakeholders.
2. Casos de usos y escenarios: Documenta la secuencia de utilización del sistema, donde los procesos están agrupados en casos de uso.
3. Requisitos del sistema: Describe detalladamente las funciones a ser desarrolladas, implementadas y provee información detallada para todos los pasos del desarrollo.

Al documentar los requisitos se recomienda hacerlo basado en tres perspectivas que se enfocan en áreas específicas del problema. Estas perspectivas no son excluyentes, por lo contrario lo recomendado es la integración de estas para garantizar la calidad del proceso [14, 16].

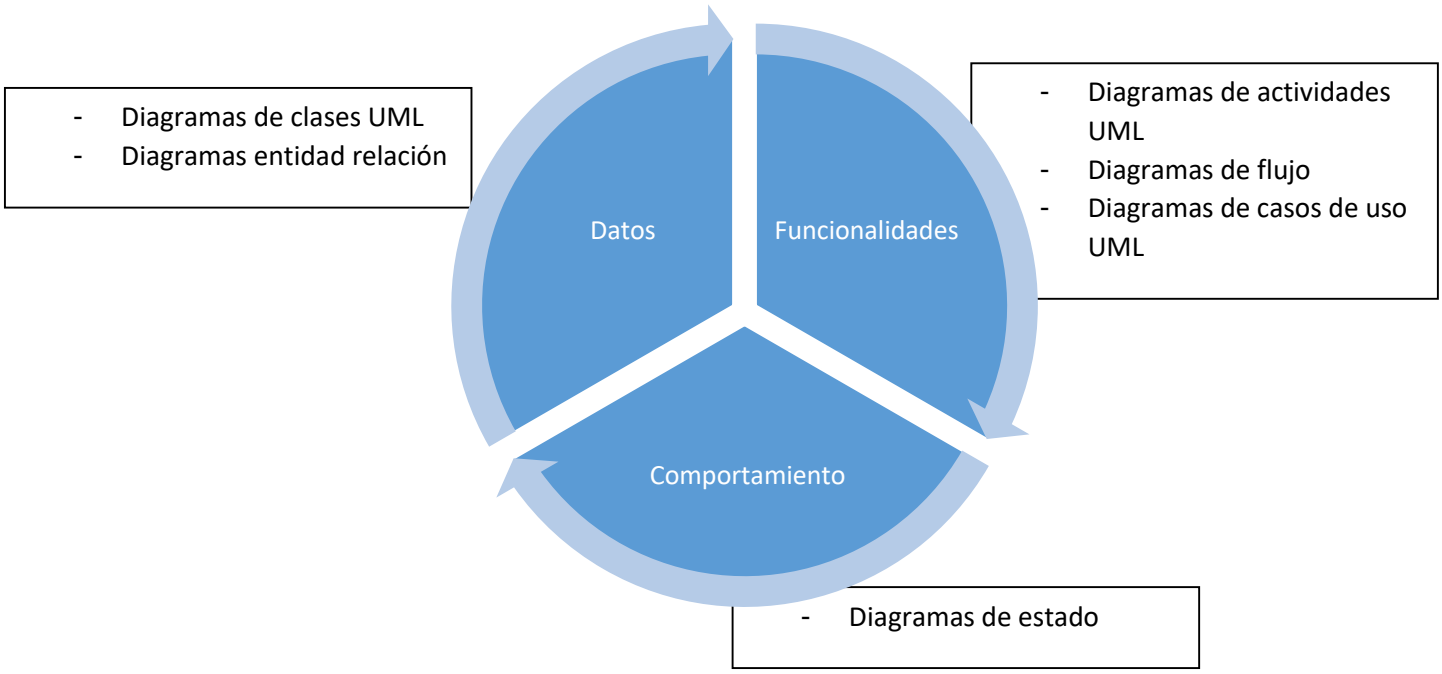


Ilustración 6 Perspectivas para el modelado de requisitos

1.3 Procesamiento de lenguaje natural

La relación existente entre el lenguaje natural y los requisitos ha constituido fuente de investigaciones desde la década de los 80 partiendo de la propuesta de utilizar los elementos sintácticos del lenguaje para el modelado de datos y el diseño de sistemas [17, 18].

Como parte del esfuerzo que se dedica en la actualidad a facilitar este proceso existen un grupo de investigaciones, que, apoyadas en los avances alcanzados en diferentes áreas del conocimiento como la minería de textos, datos y el procesamiento del lenguaje natural (PLN) entre otras que pueden ser de gran utilidad en la mejora de este proceso [19, 20].

El procesamiento del lenguaje natural es una gama de técnicas computacionales motivadas teóricamente para analizar y representar textos que ocurren naturalmente en uno o más

niveles de análisis lingüístico con el propósito de lograr un lenguaje similar al procesamiento humano para una variedad de tareas o aplicaciones [21]. En esta definición, la noción de "niveles de análisis lingüístico" se refiere a los aspectos fonético, morfológico, léxico, sintáctico, semántico, discurso y análisis pragmático del lenguaje, cuya suposición es que los seres humanos normalmente utilizan todos estos niveles para producir o comprender el lenguaje [22, 23].

El procesamiento del lenguaje natural, involucra la transformación del lenguaje natural a una representación formal, la manipulación de esa transformación. Su campo de aplicación en sumamente amplio en áreas como [24, 25]:

- Recuperación y transformación de la información.
- Traducción automática.
- Sistemas de búsqueda.
- Generación de resúmenes.
- Minería de datos.

Para el PLN se definen un conjunto de niveles, estos dependen de distintos grados de abstracción:

- Fonológico: Abarca cómo las palabras se relacionan con los sonidos que representan.
- Morfológico: Como las palabras se construyen a partir de unidades de significados más pequeñas llamadas morfemas.
- Sintáctico: Define como las palabras pueden unirse para formar oraciones, fijando el papel estructural que cada palabra juega en la oración y que sintagmas forman parte de otros sintagmas.
- Pragmático: Como se usan las oraciones en distintas situaciones y como su uso afecta el significado de las oraciones. También como el significado de las oraciones se ve afectado por las oraciones inmediatamente anteriores [26].

En la actualidad las técnicas y herramientas de procesamiento del lenguaje natural han alcanzado elevados niveles de crecimiento y poder de análisis. Sin embargo, este incremento de capacidad tiene un elevado costo en muchas situaciones, pues estos sistemas incluyen por lo general componentes de inteligencia artificial grandes volúmenes de información semántica, datos estadísticos, bases de conocimiento y algoritmos. Todo

esto con el objetivo de inferir tanta información contextual como sea posible de la fuente de origen [27, 28, 29].

1.3.1 Soluciones reportadas en la literatura para la generación automática de requisitos de software

El procesamiento del lenguaje natural es un rango de técnicas computacionales motivadas teóricamente para analizar y representar textos que ocurren naturalmente en uno o más niveles de análisis lingüístico con el propósito de lograr un procesamiento del lenguaje similar al humano para un rango de tareas o aplicaciones.

La ingeniería de requisitos soportada por procesamiento del lenguaje natural (NLP4RE) es un área de investigación y desarrollo que busca aplicar la PNL tecnologías (técnicas, herramientas y recursos) a una variedad de documentos de requisitos o artefactos para respaldar una variedad de tareas de análisis lingüístico realizadas en varias fases de ER.

Diferenciamos entre tres tipos de tecnología PNL:

Técnica PNL: Una técnica de PNL es un método, enfoque, proceso o procedimiento práctico para realizar una tarea de PNL en particular, como tokenización, segmentación (chunking), etiquetado POS, análisis sintáctico superficial y de dependencias, reconocimiento de entidades, desambiguación, análisis semántico.

Herramienta PNL: Una herramienta de PNL es un sistema de software o una biblioteca de software que admite una o más técnicas de PNL, como:

- Stanford CoreNLP7[1]: ¡CoreNLP es su ventanilla única para el procesamiento del lenguaje natural en Java!, CoreNLP permite a los usuarios derivar anotaciones lingüísticas para el texto, incluidos los límites de oraciones y tokens, partes del discurso, entidades nombradas, valores numéricos y de tiempo, análisis de dependencias y constituyentes, correferencia, sentimiento, atribuciones de citas y relaciones. CoreNLP actualmente admite 6 idiomas: árabe, chino, inglés, francés, alemán y español. La pieza central de CoreNLP es la tubería. Las canalizaciones toman texto sin formato, ejecutan una serie de anotadores de PNL en el texto y producen un conjunto final de anotaciones. Las canalizaciones producen

[1] https://stanfordnlp.github.io/CoreNLP/

CoreDocuments, objetos de datos que contienen toda la información de anotaciones, accesibles con una API simple y serializables en un búfer de protocolo de Google.

- NLTK[2]: NLTK es una plataforma líder para crear programas Python que funcionen con datos de lenguaje humano. Proporciona interfaces fáciles de usar para más de 50 corpus y recursos léxicos como WordNet, junto con un conjunto de bibliotecas de procesamiento de texto para clasificación, tokenización, derivación, etiquetado, análisis y razonamiento semántico, envoltorios para bibliotecas de PNL de nivel industrial, y un foro de discusión activo. NLTK ha sido llamado "una herramienta maravillosa para enseñar y trabajar en lingüística computacional usando Python" y "una biblioteca increíble para jugar con el lenguaje natural". El procesamiento del lenguaje natural con Python proporciona una introducción práctica a la programación para el procesamiento del lenguaje. Escrito por los creadores de NLTK, guía al lector a través de los fundamentos para escribir programas Python, trabajar con corpus, categorizar texto, analizar la estructura lingüística y más. La versión en línea del libro se ha actualizado para Python 3 y NLTK 3.
- Pattern.es: es una biblioteca para Python que contiene herramientas para la conjugación de verbos, la singularización o la pluralización de sustantivos, la división de chunks y permite realizar el etiquetado POS para español. En la versión para el idioma inglés adiciona una interfaz para WordNet. Su instalación y uso son muy sencillos. Para el etiquetado gramatical utiliza Penn TreeBank Tags. Penn TreeBank está basado en el corpus Brown, pionero en etiquetado POS para inglés. A pesar de haberse creado para el inglés, Pattern lo usa también para el español. Cada etiqueta está formada con dos o tres caracteres que indican la función que cumple cada palabra en una oración.
- SpaCy: SpaCy es una biblioteca para el procesamiento avanzado de lenguaje natural en Python. Incluye modelos estadísticos pre-entrenados y vectores de palabras, admite tokenización para más de 45 idiomas. Provee etiquetado, análisis y reconocimiento de entidades nombradas y una fácil integración de aprendizaje profundo. Es muy sencilla de utilizar, basta con incorporar desde un programa Python el modelo para el español. Devuelve un etiquetado POS completo, no sólo indica la función de la palabra en la oración, sino otros datos tales como tiempo

[2] https://www.nltk.org/

verbal, persona, número, modo, género, entre otros. La tokenizacion y el etiquetado gramatical se basan en el corpus OntoNotes57 que sigue la sintaxis de Penn TreeBank.

- Freeling[3]: Freeling es una librería de código abierto para el análisis de texto (tokenización, análisis morfológico, detección de entidades nombradas, etiquetado POS, etc.) para una variedad de idiomas. Desde su primera versión, tiene soporte para el español. Fue desarrollado por el Centro de Tecnologías y Aplicaciones del Lenguaje y el Habla (TALP) de la Universidad Politécnica de Catalunya. Se puede utilizar y ampliar los recursos lingüísticos por defecto (diccionarios, lexicones, gramáticas, etc.) adaptándolos a dominios específicos. Su código fue escrito en C++ bajo una arquitectura cliente – servidor. Existen diversas APIs que permiten utilizar Freeling en otros lenguajes, incluido Python. Su instalación no es sencilla, sobre todo en entornos Windows. Sin embargo, subsanado este inconveniente, al importar la API desde el programa Python se puede invocar a todas las funciones implementadas. El analizador morfológico de Freeling realiza el etiquetado POS usando las etiquetas EAGLES.

Recurso PNL: Un recurso de PNL es un recurso de datos lingüísticos para respaldar técnicas o herramientas de PNL, que puede ser un léxico del lenguaje (es decir, un diccionario) o un corpus (es decir, una colección de textos). Los léxicos existentes incluyen WordNet10 y FrameNet11, mientras que los ejemplos de corpus incluyen British National Corpus12 y Brown Corpus13.

1.4 Tecnologías de implementación

La implementación de la solución se desarrolló en el lenguaje de programación Python el cual es un lenguaje de programación multiparadigma que soporta varios paradigmas de programación como orientación a objetos, estructurada, programación imperativa y, en menor medida programación funcional. Se utilizó el marco de trabajo Django por su gran rendimiento y flexibilidad, lo que permite un desarrollo ágil y reutilizable. La herramienta de procesamiento de lenguaje natural SpaCy se utilizó por su fácil acceso y todo el potencial que posee.

[3] https://nlp.lsi.upc.edu/freeling/node/1

1.4.1 Python

Python[4] es un lenguaje de programación interpretado, orientado a objetos de alto nivel y con semántica dinámica. Su sintaxis hace énfasis en la legibilidad del código, lo que facilita su depuración y, por tanto, favorece la productividad. Ofrece la potencia y la flexibilidad de los lenguajes compilados con una curva de aprendizaje suave. Aunque Python fue creado como lenguaje de programación de uso general, cuenta con una serie de librerías y entornos de desarrollo para cada una de las fases del proceso de Data Science. Esto, sumado a su potencia, su carácter open source y su facilidad de aprendizaje le ha llevado a tomar la delantera a otros lenguajes propios de la analítica de datos por medio de Machine Learning como pueden ser SAS (software comercial líder hasta el momento). Python fue creado por Guido Van Rossum en 1991 y, como curiosidad, debe su nombre a la gran afición de su creador por las películas del grupo Monty Python. Además de librerías de herramientas científicas, numéricas, de herramientas de análisis y estructuras de datos, o de algoritmos de Machine Learning como NumPy, SciPy, Matplotlib, Pandas o PyBrain, Python ofrece entornos interactivos de programación orientados al Data Science.

1.4.2 Django (Framework)

Django[5] es un framework de desarrollo web de código abierto, escrito en Python, que respeta el patrón de diseño conocido como modelo–vista–controlador (MVC). Fue desarrollado en origen para gestionar varias páginas orientadas a noticias de la World Company de Lawrence, Kansas, y fue liberada al público bajo una licencia BSD en julio de 2005; el framework fue nombrado en alusión al guitarrista de jazz gitano Django Reinhardt. En junio de 2008 fue anunciado que la recién formada Django Software Foundation se haría cargo de Django en el futuro. La meta fundamental de Django es facilitar la creación de sitios web complejos. Django pone énfasis en el re-uso, la conectividad y extensibilidad de componentes, el desarrollo rápido. Python es usado en todas las partes del framework, incluso en configuraciones, archivos, y en los modelos de datos.

Otras características de Django son:

[4] https://www.python.org/
[5] https://www.djangoproject.com/

- Un mapeador objeto-relacional.
- Aplicaciones "enchufables" que pueden instalarse en cualquier página gestionada con Django.
- Una API de base de datos robusta.
- Un sistema incorporado de "vistas genéricas" que ahorra tener que escribir la lógica de ciertas tareas comunes.
- Un sistema extensible de plantillas basado en etiquetas, con herencia de plantillas.
- Un despachador de URLs basado en expresiones regulares.
- Un sistema "middleware" para desarrollar características adicionales; por ejemplo, la distribución principal de Django incluye componentes middleware que proporcionan cacheo, compresión de la salida, normalización de URLs, protección CSRF y soporte de sesiones.
- Soporte de internacionalización, incluyendo traducciones incorporadas de la interfaz de administración.

1.4.2.1 Comparación con otros frameworks de Python

Django vs Flask

Flask[6]

El matraz funciona mejor cuando desea aprovechar su personalización y la flexibilidad capacidades. Es la elección correcta cuando sabe que habrá más decisiones técnicas que tomar y no quiere estar limitado por las decisiones que ha tomado con Django. En este caso, puede valer la pena depender de un Flask eficientemente personalizable en lugar de Django. Flask funciona bien con aplicaciones web que se dividirán en varios servicios más pequeños que no necesitarán tanto código para lograr sus objetivos. También puede utilizar Flask para aplicaciones web simples que tienen funcionalidades limitadas. Cada proyecto de Flask tiene su propio "pila de tecnología interna" De frameworks, bibliotecas, etc. Es algo gratis para todos.

Django

Django se usa para compilar complicado, impulsado por base de datos sitios web al tiempo que libera al diseñador de los aspectos más monótonos de la codificación y el desarrollo. Si algún desarrollador desea crear sitios web de forma rápida y eficaz, debería optar por Django. Django es más adecuado si está más concentrado en el producto final,

[6] https://flask.palletsprojects.com/en/2.1.x/

especialmente si se trata de una aplicación específica como un sitio de noticias, una tienda electrónica o un blog. Este marco tiene un paquete de funcionalidades estándar que puede seleccionar y elegir. Puede utilizar Django para proyectos más pequeños en lugar de cambiar a Flask. Además, el enfoque integral de Django puede tener un impacto positivo en la seguridad de su aplicación.

Diferencias claves:

- Flask ofrece soporte para API, mientras que Django no tiene soporte para API.
- Django ofrece páginas HTML dinámicas, mientras que Flask no admite páginas HTML dinámicas.
- Django está diseñado para proyectos fáciles y simples, mientras que Flask es un marco web de Python creado para un desarrollo rápido.
- El framework Django tiene un despachador de URL basado en controller-regex. Por otro lado, el despachador de URL del marco web de Flask es una solicitud RESTful.
- Django es un marco web Full Stack, mientras que Flask es un marco WSGI. [30]

1.4.3 SpaCy

SpaCy[7] es una librería de software para procesamiento de lenguajes naturales desarrollado por Matt Honnibal y programado en lenguaje Python. Fue lanzado en febrero de 2015 estando su desarrollo activo y siendo utilizado en distintos entornos. Es software libre con Licencia MIT su repositorio se encuentra disponible en Github.

Características principales:

- *Tokenización* no destructiva.
- Compatibilidad con *tokenización alfa* para más de 65 idiomas.
- Soporte integrado para componentes de canalización entrenables, como reconocimiento de entidades nombradas, etiquetado de parte de la voz, análisis de dependencias, clasificación de texto, vinculación de entidades, entre otros.
- Modelos estadísticos para 17 idiomas.
- Aprendizaje multitarea con transformadores previamente entrenados como BERT.

[7] https://spacy.io/

1.4.3.1 Comparación con otras herramientas de NLP

La Universidad Católica de Salta realizo una comparación de herramientas de procesamiento de textos en español para Python y explica lo siguiente [31]:

De cada herramienta se tuvieron en cuenta algunos aspectos como la facilidad en su instalación, sobre todo para una persona con conocimientos básicos o nulos en programación; la necesidad de utilizar una interfaz de comunicación; la presencia de funciones específicas para segmentación y tokenización; el tipo de etiquetado POS que realizan y la implementación de lematizador para el lenguaje español. A modo de resumen se presenta un cuadro comparativo de algunas funciones básicas de procesamiento de texto y las herramientas mencionadas que fueron objeto de evaluación.

Herramienta	Código	Segmentación y tokenización	Etiquetado POS	Lematización
NLTK	Nativo Python	Si	Eagles	No
Freeling	API	Si	Eagles	Si
Pattern.es	Nativo Python	Si	Penn TreeBank	Si
Spacy	Nativo Python	Si	Penn TreeBank	Si
Stanford NLP	API	Si	Eagles	Si

En la segunda columna de la tabla se muestra la relación de las herramientas con el lenguaje Python. NLTK, Pattern.es y SpaCy se pueden instalar muy fácilmente de los repositorios de Python a través de la línea de comando. No ocurre lo mismo con Stanford NLP y Freeling. Ambos paquetes no son nativos de Python; fueron escritos en Java y C++, respectivamente. La instalación de Stanford es más sencilla que la de Freeling, aunque ambos necesitan sendos servidores que se estén ejecutando para funcionar. Sin embargo, si no se requiere de todo el potencial de ambas herramientas, existe otra opción para su uso: Stanford se puede utilizar a través de NLTK y Freeling provee un ejecutable que realiza la segmentación, la tokenización, el etiquetado POS y la lematización recibiendo un archivo como entrada. Todas las herramientas poseen funciones para segmentación y tokenización. NLTK, Freeling y Stanford NLP realizan el etiquetado POS con etiquetas Eagles mientras que Pattern.es y SpaCy lo hacen con etiquetas de Penn TreeBank. Salvo

NLTK, las demás herramientas, implementan un lematizador para el idioma español. El lematizador disponible en NLTK, WordNet, está disponible para el idioma inglés.

NLTK requiere que se hagan una serie de operaciones sobre el texto de segmentación, tokenizacion, etiquetado y Chunking antes de hacer el reconocimiento de entidades nombradas como tal, esto más control al usuario sobre el proceso. Por otro lado, en SpaCy se permite trabajar sobre un texto a nivel de entidad directamente, lo que simplifica mucho el proceso.

Conclusiones parciales

Al terminar el capítulo de estudio teórico se puede concluir:

1. El lenguaje natural constituye la principal fuente de información para la ingeniaría de requisitos, pero sus características dificultan el proceso ocasionando elevados valores en las estadísticas de fallos de los proyectos.
2. Existen numerosos estándares y formalizaciones de requisitos, pero el lenguaje natural y la utilización de modelos de forma independientes o combinadas en una estructura son las más comunes.
3. El contexto de las investigaciones relacionadas al procesamiento del lenguaje natural en ingeniería de requisitos se encuentra sumamente activo, con un elevado número de trabajos.
4. En el mercado se encuentran presentes varias herramientas de PLN como parte del proceso de IR, en este estudio se incluyeron solo aquellas a las cual se tuvo acceso por ser Open Source.
5. Se justificó el uso del lenguaje de programación Python para implementar nuestra propuesta de solución, así como de la librería de procesamiento de lenguaje natural SpaCy y el marco de trabajo a utilizar.

Referencias bibliográficas

[1] S. Group, «Chaos Report,» 2018.

[2] E. S. Calisaya, *CONSTRUCCIÓN DE UNA HERRAMIENTA PARA EL ANÁLISIS DE REQUISITOS DE SOFTWARE DESCRITOS EN LENGUAJE NATURAL,* AREQUIPA – PERÚ: UNIVERSIDAD NACIONAL DE SAN AGUSTÍN DE AREQUIPA, 2019.

[3] B. Boehm, Software Engineering Economics, Prentice Hall: Englewood Cliffs, 1981.

[4] B. Boehm, «Verifying and Validating Software Requirements and Design Specifications,» *IEEE,* vol. 1, nº 1, p. 75–88, 1984.

[5] I. Sommerville, «Software Engineering,» de *Software Engineering*, Boston, Pearson Education, Inc., publishing as Addison-Wesley, 2011, pp. 82-117.

[6] B. R. M. Roger S. Pressman, SOFTWARE ENGINEERING: A PRACTITIONER'S APPROACH, EIGHTH EDITION, EIGHTH EDITION ed., New York: McGraw-Hill Education, 2015.

[7] C. R. Klaus Pohl, Requirements Engineering Fundamentals, A Study Guide for the Certified Professional for Requirements Engineering Exam, 2nd Edition ed., Santa Barbara, CA: International Requirements Engineering Board, 2015.

[8] S. E. Institute, *CMMI for Development (CMMI-Dev) V1.2 Technical Report CMU/SEI-2006-TR-008 – ESC-TR-2006-008,* Pittsburgh, PA: Carnegie Mellon, 2006.

[9] I. O. f. Standardization, *[ISO/IEC 15504-5] An Exemplar Process Assessment Model,* Geneva, 2007.

[10] P. Kruchten, The Rational Unified Process: An Introduction, Addison-Wesley, 2001.

[11] International Organization for Standardization, *[ISO/IEC/IEEE 29148:2011] Systems and software engineering Life cycle processes – Requirements engineering,* Geneva, 2011.

[12] M. K. S. S. Mary Beth Chrissis, CMMI for Development, Guidelines for Process Integration and Product Improvement, Boston, MA: Addison-Wesley, Carnegie Mellon Software Engineering Institute (SEI) , 2011.

[13] J. R. S. Robertson, Mastering the Requirements Process, 2nd Edition ed., Upper Saddle River: Addison-Wesley, 2006.

[14] A. M. Davis, Software Requirements – Objects, Functions, and States, Englewood: Prentice Hall, 1993.

[15] A. H. F. B. Hussin Ahmed, «The Role of Natural Language Processing in Requirement Engineering,» *International Journal of Engineering & Technology,* vol. 7, pp. 168-171, 2018.

[16] G. B. F. v. d. L. K. Pohl, «Software Product Line Engineering – Foundations, Principles, and Techniques,» de *Springer*, Berlin, Heidelberg, 2005.

[17] P. P.-S. Chen, «English sentence structure and entity-relationship diagrams,» *Information Sciences,* vol. 29, pp. 127-149, 1983.

[18] R. J. Abbott, «Program design by informal English descriptions,» *Communications of the ACM,* vol. 26, pp. 882-894, 1983.

[19] R. A. G. B, «IDENTIFICATION OF REQUIREMENTS: A FOCUS BASED ON A VERB TAXONOMY,» *Theoria,* vol. 10, nº 1, pp. 67-79, 2001.

[20] L. Z. W. A. Alessio Ferrari, «NLP for Requirements Engineering: Tasks, Techniques, Tools, and Technologies,» de *IEEE/ACM 43rd International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, 2021.

[21] E. D. Liddy, «Natural Language Processing,» *Encyclopedia of Library and Information Science,* 2001.

[22] G. G. Chowdhury, «Natural language processing,» *Annual review of information science and technology,* vol. 37, pp. 51-89, 2003.

[23] A. J. S. R. V. G. M. d. R. C. José E. Párraga, «A Review of Automatic Requirements Extraction from Natural Language Text,» *Revista Antioqueña de las Ciencias Computacionales y la Ingeniería de Software (RACCIS),* vol. 8, nº 2, 2018.

[24] Z. M. K. N. S. Noor Hasrina Bakar, «Feature extraction approaches from natural language requirements for reuse in software product lines: A systematic literature review,» *The Journal of Systems and Software.*

[25] N. L. A. R. a. P. R. Américo Sampaio, «Mining Aspects in Requirements,» Lancaster, UK, Computing Department, Lancaster University,, 2015.

[26] E. Liddy, «Natural Language Processing,» de *Encyclopedia of Library and Information Science*, NY, Marcel Decker, Inc, 2001.

[27] R. P. Priyanka More, «Generating UML Diagrams from Natural Language Specifications,» *International Journal of Applied Information Systems (IJAIS),* vol. 1, nº 8, 2012.

[28] L. Kof, «An Application of Natural Language Processing to Domain Modelling – Two Case Studies,» *IEEE,* 2004.

[29] V. G. Vincenzo Ambriola, «Processing Natural Language Requirements,» de *IEEE* , 1997.

[30] Cynoteck, «Cynoteck,» 2022. [En línea]. Available: https://cynoteck.com/es/blog-post/flask-vs-django/. [Último acceso: 2022].

[31] A. C. M. A. Lorena Talamé, «Comparación de herramientas de procesamiento de textos en español extraídos de una red social para Python,» *ASAI, Simposio Argentino de Inteligencia Artificial.*